Volume 22 Number 2

ACTA CYBERNETICA

Szeged, 2015

ACTA CYBERNETICA

Information for authors. Acta Cybernetica publishes only original papers in the field of Computer Science. Manuscripts must be written in good English. Contributions are accepted for review with the understanding that the same work has not been published elsewhere. Papers previously published in conference proceedings, digests, preprints are eligible for consideration provided that the author informs the Editor at the time of submission and that the papers have undergone substantial revision. If authors have used their own previously published material as a basis for a new submission, they are required to cite the previous work(s) and very clearly indicate how the new submission offers substantively novel or different contributions beyond those of the previously published work(s). Each submission is peer-reviewed by at least two referees. The length of the review process depends on many factors such as the availability of an Editor and the time it takes to locate qualified reviewers. Usually, a review process takes 6 months to be completed. There are no page charges. An electronic version of the puplished paper is provided for the authors in PDF format.

Manuscript Formatting Requirements. All submissions must include a title page with the following elements:

- title of the paper
- author name(s) and affiliation
- name, address and email of the corresponding author
- An abstract clearly stating the nature and significance of the paper. Abstracts must not include mathematical expressions or bibliographic references.

References should appear in a separate bibliography at the end of the paper, with items in alphabetical order referred to by numerals in square brackets. Please prepare your submission as one single PostScript or PDF file including all elements of the manuscript (title page, main text, illustrations, bibliography, etc.). Manuscripts must be submitted by email as a single attachment to either the most competent Editor, the Managing Editor, or the Editor-in-Chief. In addition, your email has to contain the information appearing on the title page as plain ASCII text. When your paper is accepted for publication, you will be asked to send the complete electronic version of your manuscript to the Managing Editor. For technical reasons we can only accept files in LaTeX format.

Subscription Information. Acta Cybernetica is published by the Institute of Informatics, University of Szeged, Hungary. Each volume consists of four issues, two issues are published in a calendar year. Subscription rates for one issue are as follows: 5000 Ft within Hungary, €40 outside Hungary. Special rates for distributors and bulk orders are available upon request from the publisher. Printed issues are delivered by surface mail in Europe, and by air mail to overseas countries. Claims for missing issues are accepted within six months from the publication date. Please address all requests to:

Acta Cybernetica, Institute of Informatics, University of Szeged
P.O. Box 652, H-6701 Szeged, Hungary
Tel: +36 62 546 396, Fax: +36 62 546 397, Email: acta@inf.u-szeged.hu

Web access. The above informations along with the contents of past issues are available at the Acta Cybernetica homepage http://www.inf.u-szeged.hu/actacybernetica/ .

Helmut Jürgensen
Department of Computer Science
Middlesex College
The University of Western Ontario
London, Canada
hjj@csd.uwo.ca

Zoltan Kato
Department of Image Processing
and Computer Graphics
Szeged, Hungary
kato@inf.u-szeged.hu

Alice Kelemenová
Institute of Computer Science
Silesian University at Opava
Opava, Czech Republic
Alica.Kelemenova@fpf.slu.cz

László Lovász
Department of Computer Science
Eötvös Loránd University
Budapest, Hungary
lovasz@cs.elte.hu

Gheorghe Păun
Institute of Mathematics of the
Romanian Academy
Bucharest, Romania
George.Paun@imar.ro

András Prékopa
Department of Operations Research
Eötvös Loránd University
Budapest, Hungary
prekopa@cs.elte.hu

Arto Salomaa
Department of Mathematics
University of Turku
Turku, Finland
asalomaa@utu.fi

László Varga
Department of Software Technology
and Methodology
Eötvös Loránd University
Budapest, Hungary
varga@ludens.elte.hu

Heiko Vogler
Department of Computer Science
Dresden University of Technology
Dresden, Germany
Heiko.Vogler@tu-dresden.de

Gerhard J. Woeginger
Department of Mathematics and
Computer Science
Eindhoven University of Technology
Eindhoven, The Netherlands
gwoegi@win.tue.nl

In Memory of
Professor Ferenc Gécseg

Professor Ferenc Gécseg, member of the Hungarian Academy of Sciences and Professor Emeritus of the University of Szeged, passed away on 6th October, 2014.

Professor Gécseg studied at the József Attila University from 1957 to 1962 and received his master degree as mathematics teacher for secondary schools, specialising in algebra. He joined the faculty of the University of Szeged (or Attila József University as it was called at that time) in 1962 and worked at this university for his entire career. In 1976 he received the degree of Doctor of Mathematical Sciences of the Hungarian Academy of Sciences and was promoted to full professor in 1977. For several years, he was the chair of the Department of Computer Science of the Institute of Mathematics and the head of the Research Group on Theory of Automata of the Hungarian Academy of Sciences. Later he worked as full professor at the Department of Computer Algorithms and Artificial Intelligence of the Institute of Informatics. He was a founding member of the Doctoral School of Informatics.

At the beginning of his career, Professor Gécseg conducted research in universal algebra and later turned to the study of a basic algebraic model of computation, the so-called finite automaton. He obtained several results opening new vistas in the field of describing the behaviour of finite automata and systems of finite automata. His results concerning the composition and decomposition of automata play an important role, among others, in the study of the complexity of digital networks. At the end of the 1970's, he became interested in tree automata and tree transducers. He co-authored the first monograph dealing with tree automata. Since tree transducers are a mathematical model of syntax directed translation, his studies in this area are also relevant to the theory of machine translation. Professor Gécseg published three monographs, a book chapter, a textbook, and over eighty academic articles in leading periodicals and refereed conference proceedings.

He was the main organiser of the conferences 'Algebraic Theory of Automata' in the mid 1970's in Szeged which at that time played an important role in creating and maintaining connections between the scientists of our region and those of the western countries. Thanks to his teaching and research activities, an automata theory school evolved in Szeged. Several of his former students are now leading tutors and professors with extensive international contacts and quite a few of them were Széchenyi Professor Scholarship holders. A number of scientists have joined his research work both in Hungary and abroad, the number of citations for his publications is over 1000. In recognition of his academic work, he was elected corresponding member of the Hungarian Academy of Sciences in 1987, foreign member of the Finnish Academy of Sciences in 1994, and full member of the Hungarian Academy of Sciences in 1995. In 1989, he became vice-president of the European Association for Theoretical Computer Science and was reelected for another five years in 1994. He was a member of the editorial board of foreign and Hungarian periodicals and served as the editor in chief of the journal Acta Cybernetica for about two decades. He was regularly invited to programme committees of international conferences and acted as the chairman of the programme committees of FCT

81, FCT 89 and ICALP 95, which is one of the most highly ranked international conferences in theoretical computer science. These conferences were all organised in Szeged.

On several occasions he spent quite some time at foreign universities. He was a visiting professor for a year at Turku University in 1973-74; spent six months at Tampere University of Technology in 1978 and six months at the University of Western Ontario in 1987. In 1992, he worked for six months in Turku as a research professor of the Finnish Academy of Sciences.

He actively participated in the university administration and the academic public life. He was the dean of the Faculty of Science of József Attila University from 1987 to 1990. He served as a member of several ministerial and academic committees, including the Committee on Mathematics and the Committee on Information Science of the Hungarian Academy of Sciences.

Professor Gécseg played a decisive role in establishing a recognised degree programme in Informatics at the University of Szeged, in the creation of a school of fundamental research in the theory of automata in Hungary, and by raising it to international standing. At the same time, he did an exemplary work in teaching at our university and produced very high quality academic results.

His death is a great loss to the University of Szeged and the scientific community.

Szeged, September, 2015

Zoltán Ésik and Zoltán Fülöp

Acta Cybernetica **22** (2015) 247–257.

Two-Step Simulations of Reaction Systems by Minimal Ones

Arto Salomaa*

Abstract

Reaction systems were introduced by Ehrenfeucht and Rozenberg with biochemical applications in mind. The model is suitable for the study of *subset functions*, that is, functions from the set of all subsets of a finite set into itself. In this study the number of *resources* of a reaction system is essential for questions concerning generative capacity. While all functions (with a couple of trivial exceptions) from the set of subsets of a finite set S into itself can be defined if the number of resources is unrestricted, only a specific subclass of such functions is defined by *minimal* reaction systems, that is, the number of resources is smallest possible. On the other hand, minimal reaction systems constitute a very elegant model. In this paper we *simulate* arbitrary reaction systems by minimal ones in two derivation steps. Various techniques for doing this consist of taking names of reactions or names of subsets as elements of the background set. In this way also subset functions not at all definable by reaction systems can be generated. We follow the original definition of reaction systems, where both reactant and inhibitor sets are assumed to be nonempty

Keywords: reaction system, reactant, inhibitor, minimal resources, subset function, sequence

1 Introduction

A formal model of *reaction systems* was introduced by Ehrenfeucht and Rozenberg in [3]. Everything is defined within a fixed finite *background set* S. The original purpose was to model interactions between biochemical reactions. The reference [3] contains some of the original motivation and initial setup. Each reaction is characterized by its set of reactants, each of which has to be present for the reaction to take place, by its set of inhibitors, none of which is allowed to be present, and by its set of products, each of which will be present after a successful reaction. Thus, a single reaction is based on facilitation and inhibition.

*Turku Centre for Computer Science. Joukahaisenkatu 3–5 B, 20520 Turku, Finland. E-mail: asalomaa@utu.fi

Reaction systems provide a new kind of mechanism for generating functions and sequences over a finite set. A reaction system produces (in a way explained below) another subset Y of S and, thus, we have a *subset function* from subsets of S to subsets of S. Iterating the function we get a *sequence* of subsets of S. The theory of subset functions has been studied from many points of view, [15], and is particularly important in many-valued logic,[8].

Many variants of reaction systems have been introduced. The reference [1] constitutes a survey. However, the very active research in this area opens frequently new vistas. We refer to [4] for quite new developments.

Apart from various applications, reaction systems as such have been objected to many theoretical studies, [2, 5, 9, 10, 11, 12, 13]. The arising problems are mathematically very interesting, since the model is simple and clean.

However, in this paper we are concerned with the basic variant only and follow the original definition.

The elements of the sets of reactants and inhibitors are also referred to as *resources* of the reaction. Since both sets are by definition nonempty and disjoint, the smallest possible cardinality of the resource set equals 2. Such *minimal* reaction systems constitute a very simple and interesting model of computation. The class of subset functions defined by minimal reaction systems was characterized in [2]. As to be expected, the class is much smaller than the one defined by arbitrary reaction systems.

In this paper we try to narrow the gap between the generative capacities of arbitrary and minimal reaction systems. The method used below is a two-step simulation. Starting with an arbitrary reaction system, we construct a minimal one such that an arbitrary sequence of the former can be read from a sequence of the latter by taking every second subset: first, third, fifth,... The remaining subsets (second, fourth,...) contain only "junk" elements outside the background set of the original reaction system.

The exposition in this paper is largely self-contained. In particular, the basic definitions concerning reaction systems are given in Section 2. We define only the core apparatus, and do not enter additions such as a sequence of inputs from the environment, [3, 1].

2 Definitions and earlier results

We begin by defining the basic notions.

Definition 1. *A* reaction *over the finite nonempty* background set S *is a triple*

$$\rho = (R, I, P),$$

where R, I *and* P *are* nonempty *subsets of* S *such that* R *and* I *do not intersect. The three sets are referred as* reactants, inhibitors *and* products*, respectively. A* reaction system $\mathcal{A}_S$ *over the background set* S *is a finite nonempty set*

$$\mathcal{A}_S = \{\rho_j |\ 1 \leq j \leq k\},$$

of reactions over S.

In this paper S will always denote the background set. It is nonempty and finite. By *subset functions* we mean functions mapping the set 2^S into itself.

We will follow the original definition in [3] (motivated by biochemical considerations) and assume that both of the sets R and I are nonempty. It is also of definite interest to develop the theory without this assumption. This gives rise to many interesting constructions, also concerning stepwise simulation, see [6].

We will omit the index S from $\mathcal{A}_S$ whenever S is understood. The *cardinality* of a finite set X is denoted by $\sharp X$. The *empty set* is denoted by $\emptyset$. We now indicate how reactions and reaction systems are used to define subset functions.

Definition 2. *Consider a reaction $\rho = (R, I, P)$ over S and a subset T of S. The reaction ρ is* enabled *with respect to T (or for T), in symbols $en_\rho(T)$, if $R \subseteq T$ and $I \cap T = \emptyset$. If ρ is (resp. is not) enabled, then we define the* result *by*

$$res_\rho(T) = P \text{ (resp. } = \emptyset).$$

For a reaction system $\mathcal{A} = \{\rho_j |\ 1 \leq j \leq k\}$, we define the result by

$$res_{\mathcal{A}}(T) = \bigcup_{j=1}^{k} res_{\rho_j}(T).$$

An important fact to notice is that, according to Definition 2, an element in the set T is not "consumed" in the application of a reaction but is also available for other reactions when $res_{\mathcal{A}}(T)$ is computed. In the sequel we often refer to $res_{\mathcal{A}}$ as the *function defined by the reaction system $\mathcal{A}$*.

Elements in the set $R \cup I$ are also referred to as *resources*. Reaction systems are classified according to the maximal cardinality of the set of resources. We have $\sharp(R \cup I) \geq 2$, since the sets R and I are nonempty and disjoint. A reaction system is *minimal* if $\sharp(R \cup I) = 2$ holds for every reaction in the system. There is much research concerning minimal reaction systems, for instance, see [2, 7, 9, 10, 12, 13, 14]. The capacity of minimal reaction systems for defining subset functions is limited. We now quote the following fundamental result from [2], where the capacity is characterized.

Definition 3. *A subset function f is*

- union-subadditive *if $f(X \cup Y) \subseteq f(X) \cup f(Y)$,*
- intersection-subadditive *if $f(X \cap Y) \subseteq f(X) \cup f(Y)$,*

for all subsets X and Y of S.

The characterization result in [2] is now given

Theorem 1. *A function defined by a reaction system is definable by a minimal reaction system if and only if it is both union-subadditive and intersection-subadditive.*

Sequences generated by reaction systems can be viewed as *iterations* of functions $res_{\mathcal{A}}$. If $res_{\mathcal{A}}(T) = T'$, we use the simple notation

$$T \Rightarrow_{\mathcal{A}} T',$$

or simply $T \Rightarrow T'$ if $\mathcal{A}$ is understood. If

$$res_{\mathcal{A}}(T_i) = T_{i+1},\ 0 \leq i \leq m-1,$$

we write briefly

$$T_0 \Rightarrow T_1 \Rightarrow \ldots \Rightarrow T_m$$

and call m the *length* of the sequence. Since there are only $2^{\sharp S}$ subsets of S, there is always an m such that, for some $m_1 < m$, $T_m = T_{m_1}$, or else $res_{\mathcal{A}}(T_{m-1})$ is undefined, in which case we write $T_m = \emptyset$. We say that the sequence ends with a *cycle* or *terminates*, respectively. The sets T_i are usually referred to as *states* of the sequence.

Maximally inhibited reaction systems, [9, 14], offer possibilities of constructing arbitrary sequences or cycles.

Definition 4. *A reaction system with the background set S is* maximally inhibited *if every one of its reactions is of the form $(R, S - R, P)$.*

Clearly, for every reaction system $\mathcal{A}$, a maximally inhibited reaction system $\mathcal{A}'$ can be constructed such that, for any T,

$$res_{\mathcal{A}}(T) = res_{\mathcal{A}'}(T).$$

If $res_{\mathcal{A}}(T) = \emptyset$, then there is no reaction in $\mathcal{A}'$, where T is the set of reactants.

3 Names of reactions as elements of the background set

We now present our main result concerning the two-step simulation of arbitrary reaction systems by minimal ones. We have earlier, [14], presented another form of a similar construction. Names of reactions have been used as elements of the background set also in [5]. Our result shows that if one starts with a sequence (or cycle)

$$T_0 \Rightarrow T_1 \Rightarrow \ldots \Rightarrow T_m \ldots$$

according to an arbitrary reaction system $\mathcal{A}_S$, then a minimal reaction system $\mathcal{A}_M$ with the sequence

$$T_0 \Rightarrow U_0 \Rightarrow \ldots T_1 \Rightarrow U_1 \Rightarrow T_2 \ldots$$

can be constructed. The background set S_M of $\mathcal{A}_M$ includes S. Moreover, the intermediate states U_i contain only elements of $S_M - S$ and, thus, are analogous to nonterminals in grammars.

Theorem 2. *For every reaction system $\mathcal{A}$, a minimal reaction system $\mathcal{A}_M$ can be effectively constructed such that, whenever $T_0 \Rightarrow_{\mathcal{A}} T_1$, then $T_0 \Rightarrow_{\mathcal{A}_M} U_0 \Rightarrow_{\mathcal{A}_M} T_1$. Moreover, the set U_0 does not contain elements of the background set of $\mathcal{A}$.*

Proof. Let $\mathcal{A}$ have the background set S and the set of reactions

$$\rho_i = (R_i, I_i, P_i),\ 1 \leq i \leq k.$$

We now define the minimal reaction system $\mathcal{A}_M$. Its background set is

$$S_M = S \cup \{\rho_1, \ldots \rho_k, E\}.$$

It will be convenient to divide its reactions into three groups.

The first group consists of taking, for every $a \in S$, the reaction

$$(\{a\}, \{E\}, \{\rho_{i_1}, \ldots, \rho_{i_m}, E\},\ a \in I_{i_j},\ 1 \leq j \leq m.$$

No reaction results if a does not belong to any inhibitor set.

The second group consists of taking, for every $a \in S$, the reactions

$$(\{b\}, \{a\}, \{\rho_{j_1}, \ldots, \rho_{j_n}, E\},\ a \in R_{j_\nu},\ 1 \leq \nu \leq n,\ b \in S,\ b \neq a.$$

No reaction results if a does not belong to any reactant set.

The third group consists of reactions

$$(\{E\}, \{\rho_i\}, P_i),\ 1 \leq i \leq k.$$

If a sequence of $\mathcal{A}_M$ begins with $\emptyset$, there is nothing to prove. Thus, consider a nonempty $T \subseteq S$. We *claim* that the second state in the sequence beginning with T consists of E and the names of those reactions ρ_i for which $en_{\rho_i}(T)$ does NOT hold. Then only some reactions $(\{E\}, \{\rho_i\}, P_i)$ of the third group are enabled, namely, exactly those for which $en_{\rho_i}(T)$ holds. Consequently, the third state in the sequence equals $res_{\mathcal{A}_M}(T)$, and Theorem 2 follows.

To prove our claim, note first that E is always present in the second state, whereas no elements of S are present. We have to show that, whenever $en_{\rho_i}(T)$ does not hold, then ρ_i is present in the second state. By Definition 2, the relation $en_{\rho_i}(T)$ does not hold if and only if either

1. $a_1 \in T \cap I_i$, for some a_1, or else,
2. $a_2 \in R_i - T$, for some a_2.

Reactions in (1) (resp. in (2)) appear in the product set of the first (resp. the second) group of reactions of $\mathcal{A}_M$. Consequently, exactly those reactions ρ_i from the set $\{\rho_1, \ldots, \rho_k\}$ are missing from the second state of the sequence for which $en_{\rho_i}(T)$ holds. □

As an example consider the reaction system $\mathcal{A}$ with the background set $S = \{a, b, c\}$ and reactions

$$\rho_1 = (\{a, c\}, \{b\}, \{b\}),\ \rho_2 = (\{b, c\}, \{a\}, \{a, b\}),$$
$$\rho_3 = (\{b\}, \{a, c\}, \{a, b, c\}),\ \rho_4 = (\{c\}, \{a, b\}, \{a, c\}).$$

Now the minimal reaction system $\mathcal{A}_M$ has the background set

$$\{a, b, c, \rho_1, \rho_2, \rho_3, \rho_4, E\}$$

and reactions

$$(\{a\}, \{E\}, \{\rho_2, \rho_3, \rho_4, E\}),\ (\{b\}, \{E\}, \{\rho_1, \rho_4, E\}),\ (\{c\}, \{E\}, \{\rho_3, E\}),$$
$$(\{b\}, \{a\}, \{\rho_1, E\}),\ (\{c\}, \{a\}, \{\rho_1, E\}),\ (\{a\}, \{b\}, \{\rho_2, \rho_3, E\}),$$
$$(\{c\}, \{b\}, \{\rho_2, \rho_3, E\}),\ (\{a\}, \{c\}, \{\rho_1, \rho_2, \rho_4, E\}),\ (\{b\}, \{c\}, \{\rho_1, \rho_2, \rho_4, E\}),$$
$$(\{E\}, \{\rho_1\}, \{b\}),\ (\{E\}, \{\rho_2\}, \{a, b\}),\ (\{E\}, \{\rho_3\}, \{a, b, c\}),\ (\{E\}, \{\rho_4\}, \{a, c\}).$$

The first two steps in the sequence of $\mathcal{A}_M$ are listed below, beginning with the 6 possible nonempty proper subsets of S.

$$\{a, b\} \Rightarrow \{\rho_1, \rho_2, \rho_3, \rho_4, E\} \Rightarrow \emptyset$$
$$\{a, c\} \Rightarrow \{\rho_2, \rho_3, \rho_4, E\} \Rightarrow \{b\}$$
$$\{b, c\} \Rightarrow \{\rho_1, \rho_3, \rho_4, E\} \Rightarrow \{a, b\}$$
$$\{a\} \Rightarrow \{\rho_1, \rho_2, \rho_3, \rho_4, E\} \Rightarrow \emptyset$$
$$\{b\} \Rightarrow \{\rho_1, \rho_2, \rho_4, E\} \Rightarrow \{a, b, c\}$$
$$\{c\} \Rightarrow \{\rho_1, \rho_2, \rho_3, E\} \Rightarrow \{a, c\}$$

Our reaction system $\mathcal{A}$ is maximally inhibited and, consequently, each of the values $res_{\mathcal{A}}(T)$ can be seen directly from the reaction, where T is the set of reactants. This is not the case with our second example, where the reaction system is not maximally inhibited.

Thus, consider now consider the reaction system $\mathcal{A}$ with the background set $S = \{a, b, c, d\}$ and reactions

$$\rho_1 = (\{a, b\}, \{c\}, \{b, d\}),\ \rho_2 = (\{a\}, \{b, c\}, \{a\}),$$
$$\rho_3 = (\{a\}, \{c, d\}, \{c\}),\ \rho_4 = (\{b, c\}, \{a, d\}, \{a, c\}),$$
$$\rho_5 = (\{c\}, \{a, b\}, \{a, b, c, d\}),\ \rho_6 = (\{a, d\}, \{c\}, \{a, b, c\}).$$

By the construction of Theorem 2, the minimal reaction system $\mathcal{A}_M$ has the background set

$$\{a, b, c, d, \rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_6, E\}$$

and reactions

$$(\{a\}, \{E\}, \{\rho_4, \rho_5, E\}),\ (\{b\}, \{E\}, \{\rho_2, \rho_5, E\}),\ (\{c\}, \{E\}, \{\rho_1, \rho_2, \rho_3, \rho_6, E\}),$$
$$(\{d\}, \{E\}, \{\rho_3, \rho_4, E\}),\ (\{b\}, \{a\}, \{\rho_1, \rho_2, \rho_3, \rho_6, E\}),\ (\{c\}, \{a\}, \{\rho_1, \rho_2, \rho_3, \rho_6, E\}),$$
$$(\{d\}, \{a\}, \{\rho_1, \rho_2, \rho_3, \rho_6, E\}),\ (\{a\}, \{b\}, \{\rho_1, \rho_4, E\}),\ (\{c\}, \{b\}, \{\rho_1, \rho_4, E\}),$$
$$(\{d\}, \{b\}, \{\rho_1, \rho_4, E\}),\ (\{a\}, \{c\}, \{\rho_4, \rho_5, E\}),\ (\{b\}, \{c\}, \{\rho_4, \rho_5, E\}),$$
$$(\{d\}, \{c\}, \{\rho_4, \rho_5, E\}),\ (\{a\}, \{d\}, \{\rho_6, E\}),\ (\{b\}, \{d\}, \{\rho_6, E\}),$$
$$(\{c\}, \{d\}, \{\rho_6, E\}),\ (\{E\}, \{\rho_1\}, \{b, d\}),\ (\{E\}, \{\rho_2\}, \{a\}),\ (\{E\}, \{\rho_3\}, \{c\}),$$
$$(\{E\}, \{\rho_4\}, \{a, c\}, (\{E\}, \{\rho_5\}, \{a, b, c, d\}),\ (\{E\}, \{\rho_6\}, \{a, b, c\}).$$

The two-step simulation by $\mathcal{A}_M$ of the original reaction system $\mathcal{A}$ is exhibited in the following exhaustive list.

$$\begin{aligned}
\{a\} &\Rightarrow \{\rho_1, \rho_4, \rho_5, \rho_6, E\} \Rightarrow \{a, c\},\\
\{b\} &\Rightarrow \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_6, E\} \Rightarrow \emptyset,\\
\{c\} &\Rightarrow \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_6, E\} \Rightarrow \{a, b, c, d\},\\
\{d\} &\Rightarrow \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_6, E\} \Rightarrow \emptyset,\\
\{a, b\} &\Rightarrow \{\rho_2, \rho_4, \rho_5, \rho_6, E\} \Rightarrow \{b, c, d\},\\
\{a, c\} &\Rightarrow \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_6, E\} \Rightarrow \emptyset,\\
\{a, d\} &\Rightarrow \{\rho_1, \rho_3, \rho_4, \rho_5, E\} \Rightarrow \{a, b, c\},\\
\{b, c\} &\Rightarrow \{\rho_1, \rho_2, \rho_3, \rho_5, \rho_6, E\} \Rightarrow \{a, c\},\\
\{b, d\} &\Rightarrow \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_6, E\} \Rightarrow \emptyset,\\
\{c, d\} &\Rightarrow \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_6, E\} \Rightarrow \{a, b, c, d\},\\
\{a, b, c\} &\Rightarrow \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_6, E\} \Rightarrow \emptyset,\\
\{a, b, d\} &\Rightarrow \{\rho_2, \rho_3, \rho_4, \rho_5, E\} \Rightarrow \{a, b, c, d\},\\
\{a, c, d\} &\Rightarrow \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_6, E\} \Rightarrow \emptyset,\\
\{b, c, d\} &\Rightarrow \{\rho_1, \rho_2, \rho_3, \rho_4, \rho_5, \rho_6, E\} \Rightarrow \emptyset.
\end{aligned}$$

The following result follows directly from the proof of Theorem 2.

Corollary 1. *Assume that the background set and the set of reactions of a arbitrary reaction system $\mathcal{A}$ are of cardinalities s and k, respectively. Then a minimal reaction system $\mathcal{A}_M$ satisfying Theorem 2 can be effectively constructed such that the cardinalities of its background and reaction sets are $s+k+1$ and s^2+k, respectively.*

Corollary 1 is pleasing because it allows the extension of some results concerning computational complexity of reaction systems to minimal reaction systems. We hope to return to these matters in another context.

4 Extension to subset functions

Reaction systems provide a new formal tool of handling *subset functions*, that is, functions from 2^S into 2^S, where S is a finite set. This is an important aspect of reaction systems.

We begin with the following result.

Corollary 2. *Let $F(X)$ be a function mapping the set of all nonempty proper subsets of a finite set S into the set of all subsets of S. Then there is (effectively) a minimal reaction system $\mathcal{A}_M$ such that*

$$F(X) = res^2_{\mathcal{A}_M}(X).$$

Proof. The claim follows by Theorem 2 by starting with the maximally inhibited reaction system $\mathcal{A}$ with reactions $(X, S - X, F(X))$, where X runs through all nonempty proper subsets of the background set. If $F(X)$ is empty, the corresponding triple is not among the reactions of $\mathcal{A}_M$. □

It is not possible to extend Corollary 2 to concern functions F with $F(\emptyset) \neq \emptyset$. No reaction can produce anything nonempty from the empty set. We will now prove that this is, in fact, the only exception.

Theorem 3. *Let F be a subset function over the set S such that $F(\emptyset) = \emptyset$. There is effectively a minimal reaction system $\mathcal{A}_M$ such that, for all $X \subseteq S$,*

$$F(X) = res^2_{\mathcal{A}_M}(X).$$

Proof. We follow the proof of Theorem 2. We have to take care of the case, where the whole background set S appears as an argument of the function F. Assume first that $F(S) = \emptyset$. Then the proof of Theorem 2 works without any changes. The second state of the sequence of $\mathcal{A}_M$, starting with S, is $\{E, \rho_1, \ldots, \rho_k\}$. Thus, none of the reactions of the third group is enabled, yielding $\emptyset$.

Assume, secondly, that $F(S) \neq \emptyset$. In this case we make the following additions to the reaction system $\mathcal{A}_M$. The element ρ_S is added to the background set of $\mathcal{A}_M$. For every element $a \in S$, the element ρ_S is added to the product set of each reaction in the second group. The reaction $(\{E\}, \{\rho_S\}, F(S))$ is added to the third group. It is now easy to verify that

$$F(X) = res^2_{\mathcal{A}_M}(X),$$

holds for all $X \subseteq S$. □

As an example we consider the background set $S = \{a, b, c\}$ and the subset function F defined by

X	$\emptyset$	$\{a\}$	$\{b\}$	$\{c\}$	$\{a, b\}$	$\{a, c\}$	$\{b, c\}$	$\{a, b, c\}$
$F(X)$	$\emptyset$	$\emptyset$	$\{a, b, c\}$	$\{a, c\}$	$\emptyset$	$\{b\}$	$\{a, b\}$	$\{a, b\}$

This example is a modification of the first example in the preceding section.

We now define the minimal reaction system $\mathcal{A}_M$ as in Theorem 3. The minimal reaction system $\mathcal{A}_M$ has the background set

$$\{a, b, c, \rho_1, \rho_2, \rho_3, \rho_4, \rho_S, E\}$$

and reactions

$$(\{a\}, \{E\}, \{\rho_2, \rho_3, \rho_4, E\}),\ (\{b\}, \{E\}, \{\rho_1, \rho_4, E\}),\ (\{c\}, \{E\}, \{\rho_3, E\}),$$
$$(\{b\}, \{a\}, \{\rho_1, \rho_S, E\}),\ (\{c\}, \{a\}, \{\rho_1, \rho_S, E\}),\ (\{a\}, \{b\}, \{\rho_2, \rho_3, \rho_S, E\}),$$
$$(\{c\}, \{b\}, \{\rho_2, \rho_3, \rho_S, E\}),\ (\{a\}, \{c\}, \{\rho_1, \rho_2, \rho_4, \rho_S, E\}),$$
$$(\{b\}, \{c\}, \{\rho_1, \rho_2, \rho_4, \rho_S, E\}),\ (\{E\}, \{\rho_1\}, \{b\}),\ (\{E\}, \{\rho_2\}, \{a, b\}),$$
$$(\{E\}, \{\rho_3\}, \{a, b, c\}),\ (\{E\}, \{\rho_4\}, \{a, c\}),\ (\{E\}, \{\rho_S\}, \{a, b\}).$$

We obtain now

$$\{a, b, c\} \Rightarrow \{\rho_1, \rho_2, \rho_3, \rho_4, E\} \Rightarrow \{a, b\}.$$

It is easy to see that, for any $X \subseteq S$, the function value $F(X)$ appears as the third state in the sequence beginning with X.

5 Names of subsets as elements of the background set

We present for the sake of completeness the following result due to [7]. It uses names of subsets, rather than names of reactions, as an extension of the background set. While this approach is mathematically elegant, it leads to huge background sets. Apparently it can also not be extended to subset functions as Theorem 3.

Theorem 4. *For a maximally inhibited reaction system $\mathcal{A}$ with the background set S, there is effectively a minimal reaction system $\mathcal{A}_M$ such that, for every proper subset X of S,*

$$res_{\mathcal{A}}(X) = res^2_{\mathcal{A}_M}(X).$$

Proof. Since $\mathcal{A}$ is maximally inhibited, to test whether or not $en_\rho(X)$ for $\rho = (R, I, P)$, it suffices to test whether or not $X = R$. The proof is based on this observation.

We introduce a new symbol N_X (*name of* X), for every subset X of S. The background set of $\mathcal{A}_M$ is now $S \cup \{N_X | X \subseteq S\}$. Now we use the fact that, for any X, at most one reaction in $\mathcal{A}$ is enabled with respect to X, namely, the reaction $(X, S - X, P_X)$. As observed, to check whether or not not a reaction is enabled with respect to the first state X in a sequence, we only have to eliminate reactions whose reactant set is not X. This can be accomplished using the first group of reactions in $\mathcal{A}_M$

$$(\{a\}, \{b\}, \{N_X\}),\ X \subseteq S,\ a \in S - X,\ \text{or } b \in X,\ a \neq b.$$

These reactions produce the name N_Y if and only if $Y \neq X$. This means that only the correct N_X is missing from the second state of the sequence.

The second group of reactions in $\mathcal{A}_M$ consists of the following reactions

$$(\{N_Y\}, \{N_X\}, P_X),\ X, Y \subseteq S,\ X \neq Y,\ X \neq S, \emptyset.$$

These reactions yield the set P_X as the third state. □

Considering sequences, every second state consists of "junk" elements and is of size $2^s - 1$, where s is the cardinality of S.

6 Conclusion

Functions defined by minimal reaction system were characterized in [2]. However, the conditions of the characterization are very hard to test. Therefore, methods of transition from arbitrary reaction systems to minimal ones should be investigated. In this paper we have investigated the method of stepwise simulation. It is conceivable that better results are obtained by a different choice of objects whose names are used in the construction. It is a general open problem to investigate the gap between the class of functions defined by minimal and almost minimal reaction systems (where the cardinality of the resource set is at most 3 in every reaction). We conjecture that this gap is bigger than the corresponding gap between almost minimal and general reaction systems.

References

[1] R. Brijder, A. Ehrenfeucht, M. Main and G. Rozenberg, A tour of reaction systems. *International Journal of Foundations of Computer Science* 22 (2011) 1499–1517.

[2] A. Ehrenfeucht, J.Kleijn, M. Koutny and G. Rozenberg, Minimal reaction systems. *Springer Lecture Notes in Computer Science*, Vol. 7625 (2012) 102–122.

[3] A. Ehrenfeucht and G. Rozenberg, Reaction systems. *Fundamenta Informaticae* 7 (2007) 263–280.

[4] A. Ehrenfeucht and G. Rozenberg, Zoom structures and reaction systems yield exploration systems. *International Journal of Foundations of Computer Science*, 25 (2014) 275–305.

[5] E. Formenti, L. Manzoni and A.E. Porreca, On the complexity of occurrence and convergence problems for reaction systems. *Natural Computing*, DOI 10.1007/s11047-014-9456-3.

[6] L. Manzoni, D. Poças and A.E. Porreca, Simple reaction systems and their classification. *International Journal of Foundations of Computer Science* 25 (2014) 441–457.

[7] F. Montagna and G. Rozenberg, On minimal reaction systems, forthcoming. (G. Rozenberg, personal communication.)

[8] A. Salomaa, Composition sequences of functions over a finite domain. *Theoretical Computer Science* 292 (2003) 263–281.

[9] A. Salomaa, On state sequences defined by reaction systems. *Springer Lecture Notes in Computer Science*, Vol. 7230 (2012) 271–282.

[10] A. Salomaa, Functions and sequences generated by reaction systems. *Theoretical Computer Science* 466 (2012) 87–96.

[11] A. Salomaa, Functional constructions between reaction systems and propositional logic. *International Journal of Foundations of Computer Science* 24 (2013) 147–159.

[12] A. Salomaa, Minimal and almost minimal reaction systems. *Natural Computing* 12 (2013) 369–376.

[13] A. Salomaa, Compositions of reaction systems. *Journal of Automata, Languages and Combinatorics* 19 (2014) 279–290.

[14] A. Salomaa, Minimal reaction systems defining subset functions, *Springer Lecture Notes in Computer Science* 8808 (2014) 436–446.

[15] P. Zhu, h.Xie and Q. Wen, Unified definition of consistent functions. *Fundamenta Informaticae* 135 (2014) 331–340.

Dedication

This paper is dedicated to the memory of *Ferenc Gécseg*, a great scientist and a close personal friend of mine. Although the topic of the paper is not related to his work, I still consider the topic appropriate because Ferenc was always interested in new ideas and notions.

I met Ferenc in a conference arranged by him in Szeged in August 1973. We immediately shared common interests in automata theory. Besides, he was very friendly and helped with my wife's health problem as well as with our minor car accident. This was the beginning of a vigorous scientific cooperation and close friendship that extended to concern also our families. Eventually we started to call each other *Super-Brothers*. Our children met frequently, and so did some of our grandchildren. However, one cannot have everything. Our planned family meeting, with all children and grandchildren, never materialized.

Ferenc and his family spent in Turku the academic year 1974-75. The scientific cooperation, started then, continued for decades and extended to concern many of our students and colleagues. It took the form of joint papers and books, mutual visits, and conferences jointly organized. Ferenc spent several longer periods in Turku and I paid some twenty visits to Szeged. Ferenc became a member of Finland's Academy of Sciences. We both had high organizational positions in EATCS, the European Association for Theoretical Computer Science.

I did not meet Ferenc after my visit to Szeged in August 2007, but we still corresponded very much after that. Sadly I followed his deteriorating health.

Dear Super-Brother Feri, I miss you deeply. So do my family and all your friends in Finland.

Received 12th February 2015

Acta Cybernetica **22** (2015) 259–292.

Methods for Relativizing Properties of Codes

Helmut Jürgensen, Lila Kari, and Steffen Kopecki*

Abstract

The usual setting for information transmission systems assumes that all words over the source alphabet need to be encoded. The demands on encodings of messages with respect to decodability, error-detection, etc. are thus relative to the whole set of words. In reality, depending on the information source, far fewer messages are transmitted, all belonging to some specific language. Hence the original demands on encodings can be weakened, if only the words in that language are to be considered. This leads one to relativize the properties of encodings or codes to the language at hand.

We analyse methods of relativization in this sense. It seems there are four equally convincing notions of relativization. We compare those. Each of them has their own merits for specific code properties. We clarify the differences between the four approaches.

We also consider the decidability of relativized properties. If P is a property defining a class of codes and L is a language, one asks, for a given language C, whether C satisfies P relative to L. We show that in the realm of regular languages this question is mostly decidable.

In memory of Ferenc Gécseg, eminent scientist and dear friend

1 Codes in Information Systems

In an information system, a source $\mathcal{S}$ generates messages[1] which, after some modifications, enter a channel $\mathcal{K}$. The channel may change a message because of physical errors or human interference or other reasons. For a given channel $\mathcal{K}$, and an input message w, let $\kappa(w)$ be the corresponding set of potential output messages. Assume the output of the source is a message u and the corresponding input to the channel is a message $\gamma(u)$; then, as the output of the channel one may observe any message v in the set $\kappa(\gamma(u))$. The output of the channel undergoes changes again, resulting in $\delta(v)$, with the aim to recover the message originally sent as closely as possible. The technical details of this model are complicated [14]. Such details are

*Department of Computer Science, The University of Western Ontario, London, Ontario, N6A 5B7

[1] On purpose we keep the notion of "message" and much of the other entities involved at an intuitive level. A formal treatment is found in [14]. Those details would be important for the detailed picture, but do not help with the main ideas.

provided in [14, 22]; instead, we explain the concepts and ideas intuitively only. We ask the reader not to make any assumptions beyond what is being stated as those might be quite misleading.

Coding theory in general assumes that a source can generate any sequence of output symbols, albeit with differing probabilities. In reality, a source may only generate a subset M of the set of all possible output sequences[2]. For instance, a source might generate exactly the grammatically correct sentences of a given natural language. For coding theory this changes important parts of the task. Instead of the set of all potential messages one only needs to deal with the messages in M: encode these messages and decode their channel outputs into messages in M.

Thus, suppose the source generates a message u in the set M. Technical modifications, which may include compression, encryption, encoding, and even modulation change the message u into the *sent message* $\gamma(u)$. This is what enters the channel. As output of the channel one finds a *received message* $v \in \kappa(\gamma(u))$ which may differ from $\gamma(u)$ due the physical characteristics of the channel $\mathcal{K}$. From v one tries to reconstruct a message $\delta(v) = u'$ such that $u' \in M$ and, ideally, such that $u' = u$. Given the characteristics of $\mathcal{S}$ and $\mathcal{K}$, the general goal is to find γ and δ such that the whole system works well, whatever this may mean concretely[3]. The choice of γ and δ implicitly depends on the set M.

In general we assume that all entities in the model use discrete signals and synchronized discrete time[4]. In particular this means that there are finite non-empty alphabets Θ and Σ such that the messages potentially issued by the source $\mathcal{S}$ form a language $M \subseteq \Theta^+$, where Θ^+ is the set of all (non-empty, finite) words which can be formed using the letters in Θ. Σ is the set of input symbols for $\mathcal{K}$ such that $\gamma(\Theta^+) \subseteq \Sigma^+$, where Σ^+ is the set of all non-empty words over Σ. Here γ need not be a mapping, but could be a relation $\gamma \subseteq \Theta^+ \times \Sigma^+$ with $\gamma(u) = \{u' \mid (u, u') \in \gamma\}$. Σ is also the set of output symbols of the channel[5]. κ is the input-output relation of the channel. Thus $(w, v) \in \kappa$ means that v is a potential output of κ for input w. The set $\kappa(w)$ for $w \in \Sigma^+$ may contain the empty word λ, hence

$$\kappa(w) = \{v \mid (w, v) \in \kappa\} \subseteq \Sigma^* = \Sigma^+ \cup \{\lambda\}.$$

In this setting δ is a partial mapping of Σ^* into Θ^+ such that, ideally, $\delta(\kappa(\gamma(u))) \in M$ for $u \in M$. In this context, we say that γ and δ are *encodings* and *decodings,* respectively. In general, $C = \gamma(\Theta) \subseteq \Sigma^+$ is called the code[6] of γ.

[2] In a probabilistic setting, a threshold for the probability of a source output might determine the set M.

[3] For instance, if $\mathcal{S}$ and $\mathcal{K}$ are defined by probabilities, one may require the following: If $\mathcal{S}$ sends u and v is observed as the corresponding output, then the probability of u having been sent with v observed exceeds the probability of u' being sent when v is observed for all output messages of $\mathcal{S}$ different from u. For details of this probabilistic setting see [22]; for the corresponding combinatorial setting see [14].

[4] This latter assumption does not exclude synchronization errors on the logical level.

[5] To use an output alphabet different from Σ certainly is an option, but is just a nuisance generalization, which changes little.

[6] Thus a code is just a subset of Σ^+ without any further requirements; in much, but not all of the literature, the term 'code' implies unique decodability. This issue is dealt with later in this paper.

Ignoring many technical issues, γ encodes messages potentially sent by $\mathcal{S}$ and δ decodes received messages. The basic requirement is that $\delta(\gamma(u)) = u$ for all messages u. More subtle conditions may have to be satisfied, when errors need to be taken into account.

The successful functioning of such a system of information transmission depends very much on the properties of γ. In general we do not care about what happens to messages which will never be sent[7]. Hence, instead of considering the set Θ^* of all potential output messages over Θ, we focus on the set M of all potential (or likely) outputs of $\mathcal{S}$, but disregarding probabilities.

This simplifies the scenario: We eliminate the source $\mathcal{S}$ and the set of potential messages completely. Instead we consider a language $C \subseteq \Sigma^+$ serving as a code. The set M of potential messages is now replaced by the set $L \subseteq \Sigma^*$ of words which might have to be decoded as outputs of the channel. The precise relation between C and L will be discussed further below. Intuitively, the set $C^+ \cap L$ is the set of potential encoded messages, and L is the set potential channel outputs for these.

Finally we consider properties P of codes (or encodings) in this context. In general such a property would define the performance of an encoding in an information transmission setting such that the code itself determines properties of the encoding, for example: unique decodability; decoding delay; synchronization delay; error-detection; error-tolerance; error-correction. It turns out that such properties relativize in unexpected ways.

Obviously, when P contains a proposition of the form

$$\forall x_1, \ldots, x_n \in C \ \forall y_1, \ldots y_n \in \Sigma^+ \ \ldots,$$

replacing Σ^+ by the language L will change P. Intuitively, this is meant by relativizing properties of C to L.

With these preliminaries collected, we can state the main ideas of the present paper:

General Question. *Let X be a finite non-empty alphabet with at least two elements. Let L and C be non-empty languages over X. Let P be a property of languages.*

1. *Define what it means that C satisfies P relative to L.*
2. *With P fixed, what is the influence of L and vice versa?*
3. *Given P, C and L, can one decide whether C satisfies P with respect to L?*

To give this question a more concrete meaning, assume that P is the property of unique decodability: The set $C = \gamma(\Theta)$ is *uniquely decodable* if and only if every word in Σ^+ has at most one factorization into words in C; equivalently, C is uniquely

[7] This is similar to a key argument in the proof of Shannon's channel theorem (see [22], for example): Messages with probability 0 contribute errors of probability 0; hence we may ignore them and concentrate on the likely messages. Of course, messages with probability 0 can occur, but their influence has probability 0 too; hence, for practical purposes, they are ignored.

decodable if and only if every word in C^+ has exactly one factorization into words in C. In general one implicitly assumes that $L = \Sigma^+$ or $L = C^+$ depending on the requirement 'at most one' versus 'exactly one'. To adapt the concept of unique decodability to the information system at hand, one would postulate only that $L \subseteq \Sigma^+$ and that each word in L have at most one factorization. In this case, C is uniquely decodable relative to L.

Example 1.1. Let $\Sigma = \{a, b\}$, $L = (ab)^+$ and $C = \{a, ab, aab\}$. Then every word in L has a unique decoding with respect to C. On the other hand, the word aab has two distinct decodings. Hence C is not uniquely decodable in general, but uniquely decodable relative to L.

Remark 1.1. Let L and C be non-empty subsets of Σ^+. Every word in L has at most one factorization into words in C if and only if every word in $L \cap C^+$ has exactly one such factorization.

Indeed, as every word in $L \cap C^+$ has a factorization into words in C and every word in L has no more than one such factorization, each word in $L \cap C^+$ has exactly one factorization. Conversely, as the words in $L \setminus C^+$ do not have a factorization at all, when the words in $L \cap C^+$ have unique factorizations, then each word in L has at most one factorization.

We now extrapolate from this idea to consider general code properties P as discussed in [14]. We only consider error-free communication via the channel $\mathcal{K}$. Thus $v = \gamma(u)$. The more general situation of errors will require several additional difficult steps of relativization, for which we do not have a sufficient answer yet.

Earlier work with the intent to relativize various properties of codes includes papers by Head [9, 10, 11, 12], Mahalingam [23], and by Daley, Jürgensen, Kari, and Mahalingam [2]. In the present paper we do not so much consider special cases, but focus on the relativization technique itself.

To define a class of codes two intuitively different techniques tend to be used: an essentially combinatorial approach, based mainly on the structure of words in the language C; an information theoretical approach, in which the coding and decoding functions are prevalent. For a example, a prefix code C over the alphabet Σ can be defined as a set of words, such that no word in the set is a proper prefix of another word in that set; this is the combinatorial view. Equivalently, C is a prefix code if it is uniquely decodable with decoding delay 0, the information theoretic view. Each of these definitions may lead to an intuitively convincing relativization. When these turn out not to be equivalent, which one should one choose? How are they related?

We focus on this fundamental issue: How to relativize code properties of either kind? When do revitalizations coincide? When is the relativized property decidable?

For classes of codes we refer primarily to [14]. Further information is found in [1] and [27, 31].

Our paper is structured as follows: In the next section we introduce the notation and basic notions. Most of this is standard, and included only to make the paper self-contained. Some of the main unrelativized concepts are explained in that part

of the paper. In Section 3 we introduce and compare relativization methods. We review: (1) our approach of [2], which is based on a notion of admissibility; (2) the concepts proposed by Head [11]. This analysis leads to four essentially different, but equally well motivated, definitions of relativization. They are formally introduced in Section 3.3, where also their relationship, depending on the code property in question, is determined. Essentially, the four types of relativization arise from different views of how a code property might be violated when restricted to a set of messages smaller than Σ^+. While each of the four versions may be considered the "best" one, we only compare them, so as to understand what the respective strengths are. In Section 4 we consider decidability questions. Typically: Given C, L, P, and the type of relativization, we ask whether C is a code relative to L with property P and the given relativization method. The paper concludes with some general observations in Section 5.

There is a very important, but different, line of research which focuses on the relativization or generalization of just unique decodabilty. This traces back to work by Head and Weber [8, 30] and Harju and Karhumäki [7]. To our knowledge the most recent work in this field is a paper by Guzmán [6] and the thesis by Gümüştop [5].

2 Notation and Basic Notions

The sets of positive integers and of non-negative integers are $\mathbb{N}$ and $\mathbb{N}_0$, respectively. An alphabet is a non-empty set. To avoid trivial special cases, we assume that an alphabet has at least two elements. Throughout this paper Σ is an arbitrary, but fixed, alphabet. When required we add the assumption that Σ is finite. A word over Σ is a finite sequence of symbols from Σ; the set Σ^* of all words over Σ, including the empty word λ, is a free monoid generated by Σ with concatenation of words as multiplication. The set of non-empty words is Σ^+, that is, $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$. A language over Σ is a subset of Σ^*. For a language $L \subseteq \Sigma^*$ and $n \in \mathbb{N}_0$ let

$$L^n = \begin{cases} \{\lambda\} & \text{if n=0,} \\ L, & \text{if n=1,} \\ \{w \mid \exists u \in L\, \exists v \in L^{n-1} : w = uv\}, & \text{if n>1.} \end{cases}$$

Moreover, let

$$L^* = \bigcup_{n \in \mathbb{N}_0} L^n \text{ and } L^+ = \bigcup_{n \in \mathbb{N}} L^n.$$

If P is a property of languages, then $\mathcal{L}_P(\Sigma)$ is the set of languages L over Σ for which $P(L) = 1$, that is, $P(L)$ is true. We write $\mathcal{L}_P$ instead of $\mathcal{L}_P(\Sigma)$ when Σ is understood. *In the remainder of this paper, unless explicitly stated otherwise, all languages are assumed to be non-empty.*

Many classes of codes and related languages can be defined systematically in terms of relations on the free monoid Σ^+ or in terms of abstract dependence systems. See [14, 16, 28, 31] for details. In the present paper only the following relations

between words $u, v \in \Sigma^+$ are considered:

Property	*Definition*	Notation
u is a prefix of v:	$v \in u\Sigma^*$	$u \leq_{\mathrm{p}} v$
u is a proper prefix of v:	$v \in u\Sigma^+$	$u <_{\mathrm{p}} v$
u is a suffix of v:	$v \in \Sigma^* u$	$u \leq_{\mathrm{s}} v$
u is a proper suffix of v:	$v \in \Sigma^+ u$	$u <_{\mathrm{s}} v$
u is an infix of v:	$v \in \Sigma^* u \Sigma^*$	$u \leq_{\mathrm{i}} v$
u is a proper infix of v:	$(u \leq_{\mathrm{i}} v) \wedge (u \neq v)$	$u <_{\mathrm{i}} v$
u is an outfix of v:	$\exists u_1, u_2\, (u = u_1 u_2 \wedge v \in u_1 \Sigma^* u_2)$	$u\ \omega_{\mathrm{o}}\ v$
u is a proper outfix of v:	$(u\ \omega_{\mathrm{o}}\ v) \wedge (u \neq v)$	$u\ \omega_{\mathrm{o}}^{\neq}\ v$

We say that u is a *scattered subword* of v, and we write $u \leq_{\mathrm{h}} v$, if, for some $n \in \mathbb{N}$, there are $u_1, u_2, \ldots, u_n \in \Sigma^*$ and $v_1, v_2, \ldots, v_{n+1} \in \Sigma^*$ such that $u = u_1 u_2 \cdots u_n$ and $v = v_1 u_1 v_2 u_2 \cdots u_n v_{n+1}$. We write $u <_{\mathrm{h}} v$ to denote the fact that u is a proper scattered subword of v, that is, $u \leq_{\mathrm{h}} v$ and $u \neq v$. We say that u and v *overlap*, and we write $u\ \omega_{\mathrm{ol}}\ v$, if there is $q \in \Sigma^+$ such that $q <_{\mathrm{p}} u$ and $q <_{\mathrm{s}} v$ or vice versa. The relation ω_{ol} is symmetric. Note that a word can overlap itself.

To simplify or unify notation, we sometimes write ω_{p} instead of $\leq_{\mathrm{p}}$ and so on, for the partial orders above.

A binary relation ω on Σ^+ defines the property (predicate) P_ω of languages[8] $L \subseteq \Sigma^+$ as follows: $P_\omega(L) = 1$ if and only if, for all $u, v \in L$, one has $u \not\omega\ v$ and $v \not\omega\ u$. Clearly, if $P_\omega(L) = 1$ and $L' \subseteq L$, then $P_\omega(L') = 1$. Thus $P_\omega(L) = 1$ if and only if $P_\omega(\{u, v\}) = 1$ for all $u, v \in L$. Here the words u and v need not be distinct. This is important for the case of ω_{ol} for instance. Obviously, when ω is reflexive one has $P_\omega(L) = 0$ for every non-empty language L.

When $\omega = <_{\mathrm{p}}$ we write P_{p} instead of $P_{<_{\mathrm{p}}}$. Similarly, when $\omega = \omega_{\mathrm{ol}}$ we write P_{ol} instead of $P_{\omega_{\mathrm{ol}}}$. The predicates P_{s}, P_{i} and P_{o} are defined analogously starting from $<_{\mathrm{s}}$, $<_{\mathrm{i}}$ and $\omega_{\mathrm{o}}^{\neq}$, respectively.

For a set S, $\mathfrak{P}(S)$ is the set of all subsets of S and $\mathfrak{P}_{\mathrm{fin}}(S)$ is the set of all finite subsets of S. For $n \in \mathbb{N}$, let

$$\mathfrak{P}_{\leq n}(S) = \{T \mid T \in \mathfrak{P}(S), |T| \leq n\},\ \mathfrak{P}_{\geq n} = \{T \mid T \in \mathfrak{P}(S), |T| \geq n\}$$

and

$$\mathfrak{P}_{=n}(S) = \{T \mid T \in \mathfrak{P}(S), |T| = n\}.$$

In [14] the hierarchy of classes of codes is introduced using the systematic framework of abstract dependence systems. For the purposes of the present paper, the following simplified concepts suffice.

For the remainder of this section, we refer to [14, 31] and to sources cited there.

Let $C \subseteq \Sigma^+$. The language C is *uniquely decodable* if C^+ is a free subsemigroup of Σ^+ which is freely generated by C. A less abstract, but equivalent definition reads as follows:

[8] The predicate P_ω asserts that L has a certain property, defined by the negation of a relation. Admittedly, this is awkward, but it is inevitable for reconciling the two different equally convincing approaches.

Definition 2.1. *Let $C \subseteq \Sigma^+$ be a language over Σ, and let $w \in \Sigma^+$.*

1. *The word w is* C-decodable *if there are $n \in \mathbb{N}$ and words*

$$u_1, u_2, \ldots, u_n \in C \text{ such that } u_1 u_2 \cdots u_n = w.$$

   In this case, the pair $(n, (u_1, u_2, \ldots, u_n))$ is called a C-decoding *of w.*

2. *The language C is* uniquely decodable *if every word in Σ^+ has at most one C-decoding.*

Thus a language C is uniquely decodable, if and only if every word in C^+ has a unique C-decoding. We omit the reference to C when C is understood from the context. In the following we sometimes use parentheses to describe various C-decodings of a word. For example, if $C = \{a, ab, ba\}$, then $w = aba = (a)(ba) = (ab)(a)$ has two different C-decodings.

As every word in C^+ involves only finitely many elements of C, the language C is uniquely decodable if and only if every language in $\mathfrak{P}_{\text{fin}}(C)$ is uniquely decodable.

In the literature one finds the term "code" used in two different ways: (1) *a non-empty language not containing the empty word;* (2) *a uniquely decodable non-empty language not containing the empty word.* For the rest of this paper we adopt the second meaning. By $\mathcal{L}_{\text{code}}$ we denote the set of codes over Σ. For a regular language $C \subseteq \Sigma^+$ it is decidable whether $C \in \mathcal{L}_{\text{code}}$; for linear languages the code property is undecidable.

We now introduce some important classes of languages or codes. Further classes will be defined when they are needed. Let $C \subseteq \Sigma^+$.

For $n \in \mathbb{N}$ with $n > 1$, C is an *n-code* if every language in $\mathfrak{P}_{\leq n}(C)$ is a code. In general, an n-code is not necessarily a code. By $\mathcal{L}_{n\text{-code}}$ we denote the set of n-codes over Σ. For regular C it is decidable whether $C \in \mathcal{L}_{2\text{-code}}$. For $\mathcal{L}_{3\text{-code}}$ the corresponding problem is open. The n-codes form an infinite descending hierarchy with $\mathcal{L}_{\text{code}}$ as its lower bound.

The language C is a *prefix code* if, for all $u, v \in C$, $u \not\leq_{\text{p}} v$. It is a *suffix code* if, for all $u, v \in C$, $u \not\leq_{\text{s}} v$. It is a *bifix code* if it is both a prefix code and a suffix code. It is an *infix code* if, for all $u, v \in C$, $u \not\leq_{\text{i}} v$. It is an *outfix code* if, for all distinct $u, v \in C$, $u \not\leq_{\text{o}} v$. It is a *solid code* if it is an infix code and if, for all $u, v \in C$ not necessarily distinct, u and v do not overlap. The language C is a *hypercode* if, for all distinct $u, v \in C$, $u \not\leq_{\text{h}} v$.

By $\mathcal{L}_{\text{p}}$, $\mathcal{L}_{\text{s}}$, $\mathcal{L}_{\text{b}}$, $\mathcal{L}_{\text{i}}$, $\mathcal{L}_{\text{o}}$, $\mathcal{L}_{\text{h}}$, and $\mathcal{L}_{\text{solid}}$ we denote the sets of prefix codes, suffix codes, bifix codes, infix codes, outfix codes, hypercodes, and solid codes, respectively. The first six of these classes of codes are defined by predicates P_{p}, P_{s}, P_{b}, P_{i}, P_{o} and P_{h} on $\mathfrak{P}_{=2}(C)$. For $\mathcal{L}_{\text{solid}}$ we need $P_{\text{solid}} = P_{\text{i}} \wedge P_{\text{ol}}$ on $\mathfrak{P}_{\leq 2}(C)$. We also use the predicates P_{code} on $\mathfrak{P}_{\text{fin}}(C)$ and $P_{n\text{-code}}$ on $\mathfrak{P}_{\leq n}(C)$ defining $\mathcal{L}_{\text{code}}$ and $\mathcal{L}_{n\text{-code}}$, respectively.

For $n \in \mathbb{N}$, the language C is an *intercode of index n* if, $\Sigma^+ C^n \Sigma^+ \cap C^{n+1} = \emptyset$. The class $\mathcal{L}_{\text{inter}_n}$ of intercodes of index n is defined by a predicate P_{inter_n} on $\mathfrak{P}_{\leq 2n+1}(C)$ derivable from P_{i}. The set $\mathcal{L}_{\text{inter}_1}$ of intercodes of index 1 is exactly

the set $\mathcal{L}_{\text{comma-free}}$ of *comma-free codes.* The languages in $\mathcal{L}_{\text{inter}} = \bigcup_{n\in\mathbb{N}} \mathcal{L}_{\text{inter}_n}$ are called *intercodes.*

Lemma 2.1. (See [14, 31]) *The following inclusions hold:*

$$\mathcal{L}_{\text{p}} \cup \mathcal{L}_{\text{s}} \subsetneq \mathcal{L}_{\text{code}},\ \mathcal{L}_{\text{i}} \cup \mathcal{L}_{\text{o}} \subsetneq \mathcal{L}_{\text{b}} = \mathcal{L}_{\text{p}} \cap \mathcal{L}_{\text{s}},$$

$$\forall n\ \mathcal{L}_{\text{inter}_n} \subsetneq \mathcal{L}_{\text{inter}_{n+1}} \subsetneq \mathcal{L}_{\text{inter}} \subsetneq \mathcal{L}_{\text{b}},\ \mathcal{L}_{\text{h}} \cap \mathcal{L}_{\text{solid}} \subsetneq \mathcal{L}_{\text{h}} \subsetneq \mathcal{L}_{\text{i}} \cap \mathcal{L}_{\text{o}}$$

and

$$\mathcal{L}_{\text{h}} \cap \mathcal{L}_{\text{solid}} \subsetneq \mathcal{L}_{\text{solid}} \subsetneq \mathcal{L}_{\text{comma-free}} \subsetneq \mathcal{L}_{\text{i}}.$$

It will simplify the notation significantly and also open the prospects of considering a different set of problems if we weaken the definitions as follows: For

$$\varrho \in \{\text{p}, \text{s}, \text{b}, \text{i}, \text{o}, \text{h}, \text{solid}, \text{ol}, \text{inter}_n, n\text{-code}, \text{comma-free}\}$$

and potentially other types ϱ of language properties, P_ϱ is a predicate on $\mathfrak{P}_{\text{fin}}(C)$ in the following sense: A language $L \subseteq C$ has the property ϱ if and only if $P_\varrho(L)$ holds true, that is, $P_\varrho(L) = 1$; for $\varrho \in \{\text{p}, \text{s}, \text{b}, \text{i}, \text{o}, \text{solid}, \text{ol}\}$ we are mainly interested in situations when $|L| \leq 2$ as this leads to manageable decision properties. As a warning to the reader – we have seen this misread before – the set $\{u, v\}$ is equal to $\{u\}$ when $u = v$, that is, $\{u, v\}$ is not a pair, but a set.

3 Variations of Definitions

The definition of relativized codes given in [2] was phrased so as to capture and generalize the special definitions proposed by Head in [9, 10, 11, 12] in the more general framework of relations or predicates described in [14]. As noted in [2] these definitions differ in a subtle way.

In Sections 3.1 and 3.2, we review two natural proposals for relativizing code concepts. Abstracting from these, and considering other likely scenarios, it turns out that one has to consider at least four versions according to the phenomena by which violations of code properties could manifest themselves, each of them well motivated. These are investigated in detail in Section 3.3 as violation-freeness or admissibility of words. In Section 3.4 relativized codes are defined and inclusions between classes of relativized are proved. We compare the concepts considered in the earlier work [2, 11] to the ones introduced in the present paper in Section 3.5.

3.1 Admissibility of Words as Defined in [2]

We review the definitions and discussions of [2]. An improved general framework is proposed in Section 3.3.

Definition 3.1. *Let C be a subset of Σ^+ and let P be a predicate on $\mathfrak{P}_{\leq 2}(C)$. A word $q \in C^+$ is said to be* P-admissible for C *if the following condition is satisfied: if $q = xuy = x'u'y'$, with $u, u' \in C$ and $x, x', y, y' \in C^*$ then $P(\{u, u'\}) = 1$.*

This means that a word $q \in C^+$ is P-admissible if every two words $u, u' \in C$ appearing in C-decodings of q, together satisfy the property P. For example, for $P = P_{\mathrm{p}}$, a word q is *prefix-admissible,* if no two words $u, u' \in C$ appearing in C-decodings of q are strict prefixes of each other. There is a subtle point: Suppose that u' is a proper prefix of u. For a word q, three different situations need to be considered:

1. The word q has a C-decoding of the form
$$\cdots(u')\cdots(u)\cdots \text{ or } \cdots(u)\cdots(u')\cdots .$$
2. The word q has two C-decodings of the forms
$$\cdots(u')\cdots \text{ and } \cdots(u)\cdots .$$
3. The word q has two C-decodings of the form $q_1(u')v'q_2$ and $q_1(u)q_2$ with $u = u'v'$ where $q_1, q_2, v'q_2 \in C^*$, $v' \in \Sigma^+$.

The difference between these situations becomes apparent in our discussion of relativized solid codes below. Definition 3.1 applies to any occurrences of u and u', not just to those situations in which u and u' start at the same position in q, and also not just to occurrences of u and u' in the same C-decoding of q. Thus, if u and u' are distinct and occur in any C-decodings of a word $q \in L$, which is prefix-admissible for C, then the set $\{u, u'\}$ must be a prefix code.

Similarly, a word $q \in C^+$ is *overlap-admissible* if no two words $u, u' \in C$, not necessarily distinct and appearing in any C-decodings of q, overlap. In particular, if $u \in C$ and u occurs in a C-decoding of q, then u must not overlap itself.

Definition 3.2. *Let C be a subset of Σ^+, let $L \subseteq C^+$ and let P be a predicate on $\mathfrak{P}_{\leq 2}(C)$. Then C is said to* satisfy P relative to L *if every $q \in L$ is P-admissible for C.*

Definition 3.3. *When C satisfies P relative to L we say that C is a* P-code relative to L.

As the predicate P is arbitrary, a P-code relative to L need not be uniquely decodable even when $L = C^+$. The restriction of L being a subset of C^+ turns out to be too restrictive in the new context of this paper and is lifted starting in Section 3.3.

The following trivial observation is used without special mention in the sequel.

Remark 3.1. Let P, P_1 and P_2 be predicates on $\mathfrak{P}_{\leq 2}(C)$ with $P = P_1 \wedge P_2$. Let q, C and L be as in Definitions 3.1, 3.3 and 3.2. The following statements hold true:

1. q is P-admissible for C if and only if q is both P_1-admissible and P_2-admissible for C.
2. C satisfies P relative to L if and only if C satisfies P_1 and P_2 relative to L.
3. C is a P-code relative to L if and only if C is both a P_1-code and a P_2-code relative to L.

3.2 Definitions Inspired by Tom Head

In [11] and related papers, Head proposed various relativizations of code concepts. The most relevant for the present discussion, because it introduces issues not encountered in other contexts, is that of relativized solid codes. The formalism used here leads to a novel general concept of relativization. This section of the paper summarizes ideas and statements from [2] relevant to the issue at hand.

Definition 3.4. ([9]) *Let C and L be non-empty subsets of Σ^+. The set C is a* solid code relative to L *if it satisfies the following conditions for all words $q \in L$:*

1. *if $q = xszty$ with $x, y, s, t \in \Sigma^*$ such that $z, szt \in C$, then $st = \lambda$;*
2. *if $q = xszty$ with $x, y, s, t \in \Sigma^*$ such that $sz, zt \in C$ and $z \in \Sigma^+$ then $st = \lambda$.*

The first condition states that, for $u, v \in C$, if $u <_{\mathrm{i}} v$, then, for all $q \in L$, $v \not\leq_{\mathrm{i}} q$. The second condition states that if $u, v \in C$, and u and v overlap as $u = sz$ and $v = zt$ with $z \in \Sigma^+$, then, for all $q \in L$, $szt \not\leq_{\mathrm{i}} q$.

Definition 3.4 is one possible relativization of the notion of solid code. It differs from the notion of P_{solid}-code relative to a language as introduced in Definition 3.3.

Note that, if C is a solid code relative to L then C is a P_{i}-code relative to $L \cap C^+$. Indeed, let q in $L \cap C^+$. If $u \in C$ occurs in a C-decoding of q, $v \in C$ and $u <_{\mathrm{i}} v$, then $v \not<_{\mathrm{i}} q$. Hence v does not occur in a C-decoding of q. As shown in Example 3.1 below, C being a solid code relative to L does not imply that C is a P_{ol}-code or a P_{solid}-code relative to L.

For (unrelativized) solid codes there is also a definition based on decompositions of messages (see [14]): Let C be a subset of Σ^+ and $q \in \Sigma^+$. A *C-decomposition of q* consists of two sequences $u_0, u_1, \ldots, u_n \in \Sigma^*$ and $v_1, v_2, \ldots, v_n \in C$ for some $n \in \mathbb{N}$, such that $q = u_0 v_1 u_1 v_2 u_2 \cdots v_n u_n$ and $v \not\leq_{\mathrm{i}} u_i$ for all $v \in C$ and $i = 0, 1, \ldots, n$. Every word $q \in \Sigma^+$ has at least one C-decomposition. Note that every C-decomposition of a word in C^+ can be considered as a C-decoding as follows:

$$u_0 = u_1 = \cdots = u_n = \lambda$$

and the C-decoding is

$$(n, (v_1, v_2, \ldots, v_n)).$$

The set C is a solid code if and only if every word in Σ^+ has a unique C-decomposition. In [13], a relativization of the notion of solid code is proposed, which is based on the uniqueness of C-decompositions, and this notion turns out to be equivalent to the one of Definition 3.4.

Proposition 3.1. ([13]) *Let $L \subseteq \Sigma^+$. A language $C \subseteq \Sigma^+$ is a solid code relative to L if and only if every word $q \in L$ has a unique C-decomposition.*

The difference between these equivalent concepts and our approach to relativizing solid codes is illustrated by the following example.

Example 3.1. ([11]) Let $\Sigma = \{a, b, c\}$ and $C = \{ab, c, ba\}$. The set C is not overlap-free, hence not a solid code. By Definition 3.4, C is a solid code relative to the language $L = (\{abc\} \bigcup \{cba\})^*$. However, the set C is not a P_{solid}-code relative to L, as $q = abccba \in L$ has the C-decoding $(ab)(c)(c)(ba)$ and is thus not P_{solid}-admissible since $ab\ \omega_{\text{ol}}\ ba$.

The main differences between Definitions 3.3 and 3.4 are as follows:

1. According to Definition 3.3, the mere and possibly unrelated existence of words for which the predicate is false constitutes a violation. According to Definition 3.4, the words in question must be in a specific violating position.

2. According to Definition 3.3, the words in violation must occur in C-decodings. According to Definition 3.4, they may appear anywhere.

In the next section, we analyse these differences and provide new definitions according to the analysis. Altogether, we have to investigate four different ways in which code concepts can be relativized.

3.3 Violating Instances

There does not seem to be a unique best scheme for relativizing code properties. All proposed schemes seem to diverge not only when the language L relativized to is a subset of Σ^+ or of C^+, but also when the particular types of violations of the code properties are considered. We now identify four violating scenarios in very general terms. These seem to be the most common ones in real systems. For specific natural code properties we state their relativizations. We also determine the connection between the four notions of violation. Our basic definitions may seem to be far too general; this permits us to capture most of the interesting cases and to leave the field open for other cases which might require a relativization as well.

To clarify the intuition, we start with examples. We consider a language $C \subseteq \Sigma^+$ and a predicate P defining a class of codes.

A violating instance of P_{p}, the prefix-freeness predicate, would be the occurrence of a word $v \in C$ such that there is a word $u \in C$ with $u <_{\text{p}} v$, that is, $P_{\text{p}}(\{u, v\}) = 0$. For P_{s}, P_{i}, P_{h}, P_{o} and several other such predicates we have analogous characterizations. To help the readers' intuition we switch freely between predicates and relations whenever one or the other seems easier to understand.

Take P_{b}. One has $P_{\text{b}}(\{u, v\}) = 0$ if $u <_{\text{p}} v$ or $u <_{\text{s}} v$ or vice versa. Thus there are two potential violating instances of P_{b}, manifested as violating instances of P_{p} and P_{s}, respectively.

This seems to determine the pattern for predicates defined by conjunctions or disjunctions of predicates.

Thus a violating instance of the conjunction (intersection) P of two predicates P_1 and P_2 could be a violating instance of P_1 or a violating instance of P_2. Dually, if P is defined as the disjunction (union) of two such predicates, then violating

instances of P are exactly the instances which are violating both P_1 and P_2. This idea works well also with $P_{\text{solid}} = P_{\text{i}} \wedge P_{\text{ol}}$.

These considerations suggest the following tentative definition:

> *Let P be an n-ary predicate. A* violating instance *of P is a set $\{u_1, \ldots, u_n\}$ of words such that $P(\{u_1, \ldots, u_n\}) = 0$.*

This definition is not good enough as it does not capture how the words in question are actually located with respect to each other; hence, a proper definition needs to be based on relations or tuples with special properties rather than sets.

We consider a set of more detailed examples in order to detect a pattern. For

$$\varrho \in \{\text{p}, \text{s}, \text{pi}, \text{si}, \text{b}, \text{i}, \text{o}, \text{h}, \text{solid}, \text{ol}, \text{inter}_n, \text{comma-free}\}$$

and potentially other types ϱ of language properties, let ω_ϱ or $<_\varrho$ be the corresponding relation or partial order, and P_ϱ be the corresponding predicate. Let $C \subseteq \Sigma^+$.

1. A violating instance of P_{p} is the occurrence of a word $v \in C$ such that there is $u \in C$ with $u <_{\text{p}} v$. Similarly for P_{s}, P_{pi}, P_{si}, and P_{i}.

2. A violating instance of P_{b} is the occurrence of a word $v \in C$ such that there is $u \in C$ with $u\, \omega_{\text{b}}\, v$, that is, $u <_{\text{p}} v$ or $u <_{\text{s}} v$.

3. A violating instance of P_{o} is the occurrence of a word $v \in C$ such that there is $u \in C$ with $u\, \omega_{\text{o}}\, v$ and $u \neq v$.

4. A violating instance of P_{ol} is the occurrence of a word $w = w_1 w_2 w_3$ with $w_1, w_2, w_3 \in \Sigma^+$ such that $w_1 w_2 \in C$ and $w_2 w_3 \in C$; thus, $w_1 w_2\, \omega_{\text{ol}}\, w_2 w_3$.

5. A violating instance of P_{solid} is the occurrence of a word which is a violating instance of P_{i} or of P_{ol}.

6. A violating instance of P_{inter_n} is the occurrence of a word

   $$w = v_1 v_2 \cdots v_{n+1} \text{ with } v_1, v_2, \ldots, v_{n+1} \in C$$

   such that there are words

   $$u_1, u_2, \ldots, u_n \in C \text{ and } x, y \in \Sigma^+$$

   with $x u_1 u_2 \cdots u_n y = w$.

7. A violating instance of $P_{\text{comma-free}}$ is the occurrence of a word $w = v_1 v_2$ such that there are words $u \in C$ and $x, y \in \Sigma^+$ with $xuy = w$.

8. A violating instance of P_{h} is the occurrence of a word $v \in C$ such that there is a word $u \in C$ with $u <_{\text{h}} v$.

The cases (1), (2), (3), (6), (7), and (8) above have in common that the (proper) relation involved has a "direction": The relations for

$$\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}\}$$

are anti-symmetric. For $\varrho \in \{\mathrm{inter}_n, \text{comma-free}\}$, that is, cases (6) and (7), one considers the relations $\omega_{\mathrm{inter}_n}$ and $\omega_{\text{comma-free}}$ defined as follows[9] [14]:

- $\omega_{\mathrm{inter}_n}$ is a $(2n+1)$-ary relation on C such that

  $$(u_1, u_2, \ldots, u_n, v_1, v_2, \ldots, v_{n+1}) \in \omega_{\mathrm{inter}_n}$$

  if and only if there are $x, y \in \Sigma^+$ such that $v_1 v_2 \cdots v_{n+1} = x u_1 u_2 \cdots u_n y$.
- $\omega_{\text{comma-free}} = \omega_{\mathrm{inter}_n}$ for $n = 1$.

We interpret $\omega_{\mathrm{inter}_n}$ as a binary relation between n-tuples and $(n+1)$-tuples of words in C. Let $\overline{\omega_{\mathrm{inter}_n}}$ be this binary relation, that is,

$$(u_1, u_2, \ldots, u_n, v_1, v_2, \ldots, v_{n+1}) \in \omega_{\mathrm{inter}_n}$$

if and only if

$$\big((u_1, u_2, \ldots, u_n), (v_1, v_2, \ldots, v_{n+1})\big) \in \overline{\omega_{\mathrm{inter}_n}}.$$

Similarly, we obtain $\overline{\omega_{\text{comma-free}}}$ from $\omega_{\text{comma-free}}$. Then, by definition, both $\overline{\omega_{\mathrm{inter}_n}}$ and $\overline{\omega_{\text{comma-free}}}$ are anti-symmetric binary relations.

For P_{ol}, instead of considering a binary relation between code words, it seems more adequate to consider a binary relation $\overline{\omega_{\mathrm{ol}}}$ between a pair (u_1, u_2) of codewords and a word $w \in \Sigma^+$ such that $(u_1, u_2)\ \overline{\omega_{\mathrm{ol}}}\ w$ if and only if there are $w_1, w_2, w_3 \in \Sigma^+$ such that $u_1 = w_1 w_2$, $u_2 = w_2 w_3$, and $w_1 w_2 w_3 = w$.

One could apply similar modifications to the relations defining the outfix codes, the hypercodes, and all the codes in the shuffle hierarchy. For example, instead of $<_{\mathrm{h}}$ one could use the relation $\overline{\omega_{\mathrm{h}}}$ defined as follows: $(u_1, u_2, \ldots, u_k)\ \overline{\omega_{\mathrm{h}}}\ (v)$ if and only if

$$v \in \Sigma^* u_1 \Sigma^* u_2 \cdots \Sigma^* u_k \Sigma^* \text{ and } u_1 u_2 \cdots u_k \neq v,$$

where $k \in \mathbb{N}$ and $u_1, u_2, \ldots, u_k, v \in \Sigma^+$. For the present purposes the following, less intuitive, alternative

$$(u_1, u_2, \ldots, u_k)\ \overline{\omega_{\mathrm{h}}}\ (v) \text{ if and only if } u_1 u_2 \cdots u_k <_{\mathrm{h}} v$$

would also work. The former captures the idea that $u_1 \leq_{\mathrm{i}} q$, $u_2 \leq_{\mathrm{i}} q$, …, $u_k \leq_{\mathrm{i}} q$ in the order given by the k-tuple $(u_1, u_2, \ldots, u_k)$. The latter is a simple reformulation of the embedding order. For our purposes, neither modification is needed.

Note that the transition from a relation ω_ϱ to its overlined version $\overline{\omega_\varrho}$ is *ad hoc* and not claimed to be in any way defined by an operator. We introduce the latter only for convenience. In the sequel, to keep the notation simple, we drop the distinction when there is no risk of confusion. For example, a statement of the form

[9] In [14] the order of the components is different. The change is not essential, but simplifies the presentation.

"For $\varrho \in \{\mathrm{p}, \ldots, \mathrm{inter}_n, \ldots\}$ the relation ω_ϱ satisfies ..."

refers to ω_p for $\varrho = \mathrm{p}$ and, depending on the context, to $\omega_{\mathrm{inter}_n}$ or to $\overline{\omega_{\mathrm{inter}_n}}$ for $\varrho = \mathrm{inter}_n$.

To define violating instances in rather general terms, we consider binary relations on tuples of words and their corresponding binary predicates.

For any set S and any $n \in \mathbb{N}$, let $n\text{-}\mathsf{tuples}(S)$ be the set of n-tuples of elements in S and let $\mathsf{all\text{-}tuples}(S) = \bigcup_{n\in\mathbb{N}} n\text{-}\mathsf{tuples}(S)$. We consider binary relations ω between tuples of words over Σ. Typically there is a small upper bound on the arity of the tuples. Such a relation ω would be a subset of

$$\bigcup_{1\le k\le m} \bigcup_{1\le n\le m} k\text{-}\mathsf{tuples}(\Sigma^*) \times n\text{-}\mathsf{tuples}(\Sigma^*)$$

for some $m \in \mathbb{N}$. In some quite natural situations however, like that of hypercodes, there might not be *a priori* bounds on k and n. This concern will be kept in mind as we propose definitions. As such relations are defined by (disjoint) unions of relations in a natural way as expressed by the formula above, their respective properties are conjunctions of the individual properties according to the constituents. The details are explained by example below.

Definition 3.5. *Let $n \in \mathbb{N}$ and let $\mathfrak{u} = (u_1, u_2, \ldots, u_n)$ be an n-tuple of words in Σ^*. Define $\mathsf{word}(\mathfrak{u})$ as the word $u_1u_2\cdots u_n$. Moreover, for $u \in \Sigma^*$, let $\mathsf{word}(u) = u$.*

The present goal is as follows: Let $C \subseteq \Sigma^+$, $C \neq \emptyset$. For a word $q \in \Sigma^+$ we want to express that q does not contain words in C which violate a binary relation ω on $\mathsf{all\text{-}tuples}(\Sigma^+)$, the latter defining a class of languages or codes. Additionally, if $\mathfrak{u}\,\omega\,\mathfrak{v}$, then $\mathsf{word}(\mathfrak{u})$ and $\mathsf{word}(\mathfrak{v})$ must appear in some "natural" relationship within q. A first attempt towards this goal might read as follows:

> Let ω be a binary relation on $\mathsf{all\text{-}tuples}(\Sigma^+)$ and let $q \in \Sigma^+$. A *violating instance of ω in q* is a pair $(\mathfrak{u}, \mathfrak{v})$ of distinct tuples of words in Σ^+ such that $\mathfrak{u}\,\omega\,\mathfrak{v}$ and $\mathsf{word}(\mathfrak{v}) \le_\mathrm{i} q$.

At a first glance this seems to be a clean definition. It only involves the relation ω, but not the set C, and the latter can be built in later as a constraint. The following example shows that the attempted definition will not work without a connection to C.

Example 3.2. Let $\Sigma = \{a, b\}$ and $\omega = <_\mathrm{p}$. Then every word of length at least 2 contains a violating instance of $<_\mathrm{p}$.

Nevertheless, we work with this intuition. It does not lead to a general definition, but at least to a usable one for many relevant cases. To simplify terminology, when $(\mathfrak{u}, \mathfrak{v})$ is a violating instance of ω in q in the tentative sense above, we also say, equivalently, that q *contains* $(\mathfrak{u}, \mathfrak{v})$ *as a violating instance of* ω – or of the predicate P_ω defining ω.

For $\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}\}$ we just consider the relation ω_ϱ. Similarly, for the relations defining the shuffle hierarchy. For $\varrho \in \{\mathrm{inter}_n, \text{comma-free}, \mathrm{ol}\}$, the relations $\overline{\omega_{\mathrm{inter}_n}}$, $\overline{\omega_{\text{comma-free}}}$, and $\overline{\omega_{\mathrm{ol}}}$ will serve. Thus, also the solid codes are included. In each of the cases considered here, $\mathsf{word}(\mathfrak{v}) \leq_{\mathrm{i}} q$ implies that each component of $\mathfrak{u}$ is a subword, possibly scattered, of q. Our present motivation was to cover as much as possible of the code hierarchy of [14].

To address the problems with the notion of violating instance of a relation ω, we consider, simultaneously, a relation ω, a non-empty set C of words in Σ^+, and a word $q \in \Sigma^+$. The relation ω is meant to describe a class of languages – or codes – such that C does not contain any words which would lead to a violating instance in q. Without loss of generality, one can assume that ω is irreflexive. We did not find a satisfactorily simple definition which could be applied to any binary relation on $\mathsf{all\text{-}tuples}(\Sigma^+)$. Especially relations like ω_{ol} or $\overline{\omega_{\mathrm{ol}}}$ cause difficulties, as the relative positions of the occurrences of their components in a word are not fixed. Therefore, from here on we consider only a restricted class of relations:

$$\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}, \mathrm{solid}, \mathrm{ol}, \mathrm{inter}_n, \text{comma-free}\}.$$

Definition 3.6. *Let $C \subseteq \Sigma^+$ and $q \in \Sigma^+$. Let $\omega \neq \omega_{\mathrm{ol}}$ be an irreflexive, binary relation on $\mathsf{all\text{-}tuples}(\Sigma^+)$ such that, for all $\mathfrak{u}, \mathfrak{v} \in \mathsf{all\text{-}tuples}(\Sigma^+)$, $\mathfrak{u}\ \omega\ \mathfrak{v}$ implies $\mathsf{word}(\mathfrak{u}) \leq_{\mathrm{h}} \mathsf{word}(\mathfrak{v})$. Let P_ω be the predicate defining ω.*

1. *The word q is P_ω-*violation-free for decompositions *with respect to C, if there are no $\mathfrak{u}, \mathfrak{v} \in \mathsf{all\text{-}tuples}(C)$ such that $\mathfrak{u}\ \omega\ \mathfrak{v}$ and $\mathsf{word}(\mathfrak{v}) \leq_{\mathrm{i}} q$.*

2. *The word q is P_ω-*violation-free for decodings *with respect to C, if for all $q_1, q_2 \in C^*$ and all $\mathfrak{v} \in \mathsf{all\text{-}tuples}(C)$ with $q = q_1\mathsf{word}(\mathfrak{v})q_2$ there is no $\mathfrak{u} \in \mathsf{all\text{-}tuples}(C)$ such that $\mathfrak{u}\ \omega\ \mathfrak{v}$.*

3. *The word q is said to be P_{ol}-*violation-free for decompositions *with respect to C, if there are no words $u, v, w \in \Sigma^+$ such that $uv, vw \in C$ and $uvw \leq_{\mathrm{i}} q$.*

4. *The word q is said to be P_{ol}-*violation-free for decodings *with respect to C, if there are no words $q_1, q_2 \in C^*$ and $u, v, w \in \Sigma^+$ with $uv, vw \in C$ such that $q = q_1uvq_2$ with $w \leq_{\mathrm{p}} q_2$ or $q = q_1vwq_2$ with $u \leq_{\mathrm{s}} q_1$.*

To explain Definition 3.6, we consider the special cases of prefix codes, outfix codes, intercodes of some index n, and solid codes defined by the relations $<_{\mathrm{p}}$, $\omega_{\mathrm{o}}^{\neq}$, $\omega_{\mathrm{inter}_n}^{\neq}$, and ω_{solid} as characteristic examples. Most other cases in the hierarchy of codes are analogous. In the definition we attempt to capture an essential idea of Head's relativization: the respective code property is violated if and only if the words involved appear exactly in the relative positions as defined by the code property. Beyond that, we distinguish between violating instances for decompositions and violating instances for decodings. The former may occur anywhere in the word q under consideration – and this is the case of Head's definition (Definition 3.4); the latter can only occur at positions defined by a decoding. This distinction turns

out to be important, as fewer positions in a word under consideration need to be examined in the case of decodings compared to the case of decompositions.

Head's relativization of the condition of overlap-freeness in Definition 3.4 really applies only to decompositions. In Definition 3.6(4) we propose a possible interpretation of Head's approach in the context of decompositions. Another possible interpretation would be as follows.

> Let $q = q_1uvwq_2 \in C^*$ with $uv, vw \in C$ and $u, v, w \in \Sigma^+$. Then $q_1, wq_2 \in C^*$ or $q_2, q_1u \in C^*$.

This is equivalent to Definition 3.6(4).

The first two parts of Definition 3.6 refer to binary relations ω on tuples of words in Σ^+ such that $\mathfrak{u}\ \omega\ \mathfrak{v}$ implies that $\mathsf{word}(\mathfrak{u}) \leq_\mathrm{h} \mathsf{word}(\mathfrak{v})$. Thus, if $\mathsf{word}(\mathfrak{v}) \leq_\mathrm{i} q$, then $\mathsf{word}(\mathfrak{u})$ appears as a possibly scattered subword of that occurrence of $\mathsf{word}(\mathfrak{v})$ in q. The relation ω_ol is an important example of a relation which does not fit into this pattern. We include ω_ol as a special case in Definition 3.6 to exhibit unsolved problems in the relativization methods and the need for a more inclusive approach.

Among the cases for illustrating Definition 3.6, a simple one is that of prefix codes and the like. The class of codes C is defined by a partial order $\omega^{\neq}$ on Σ^+ such that $u, v \in C$ implies that $u \not\omega^{\neq} v$. Moreover, $u\ \omega^{\neq}\ v$ implies $u <_\mathrm{i} v$. If q contains a violation of ω, then there are $u, v \in C$ such that $u\ \omega^{\neq}\ v$ and $v \leq_\mathrm{i} q$. Thus the mere occurrence of v as an infix of q results in a violating instance, for decompositions. For decodings, the word v has to appear at a special spot, determined by a decoding; but note that the decoding need not be unique.

The case of outfix codes and of all shuffle codes down to the class of hypercodes requires special consideration as to what we mean by "violation". The case of outfix codes is indicative of the issues. Suppose u is a proper outfix of v. Then $v = u_1v_0u_2$ with $u_1u_2 \in \Sigma^+$, $u = u_1u_2$, and $v_0 \in \Sigma^+$. If $v \leq_\mathrm{i} q$ we have a violating instance according to the definition, but $u \not\leq_\mathrm{i} q$. Do we want this? We argue as follows: The intent of using an outfix code could be to detect insertion errors, like the ones which change u into v. In this, clearly, the occurrence of v in q gives rise to an ambiguity as to how q should be read (both for decompositions and decodings). Similar arguments concern the whole shuffle hierarchy and motivate the condition of $\mathsf{word}(\mathfrak{u}) \leq_\mathrm{h} \mathsf{word}(\mathfrak{v})$ above. In general, the embedding is completely determined by ω.

The case of intercodes of index n is special only in that we deal with tuples of words. The relation defining the intercodes satisfies the conditions trivially.

Finally, for solid codes we need to consider the relation $\omega_\mathrm{solid} = <_\mathrm{i} \cup \omega_\mathrm{ol}$. The rôle of $<_\mathrm{i}$ is similar to that of the prefix order above. The rôle of ω_ol is different. Regardless of whether we use ω_ol or $\overline{\omega_\mathrm{ol}}$, there is a problem which seems to require special measures.

- Using ω_ol: If $u\ \omega_\mathrm{ol}\ v$ with $u, v \in \Sigma^+$ then $v \leq_\mathrm{i} q$ does not imply that $u \leq_\mathrm{h} q$.
- Using $\overline{\omega_\mathrm{ol}}$: If $(u_1, u_2)\ \overline{\omega_\mathrm{ol}}\ v$ then $v \leq_\mathrm{i} q$ does not imply $\mathsf{word}((u_1, u_2)) = u_1u_2 \leq_\mathrm{h} q$. However, we have $u_1 \leq_\mathrm{i} q$ and $u_2 \leq_\mathrm{i} q$.

In either case, the mere occurrence of v does not result in a violating instance in general.

Example 3.3. Let $\Sigma = \{a, b\}$.

1. Consider the prefix order $<_{\mathrm{p}}$ and the language $C = \{a, ab\}$. This language is not a prefix code. The set of words which are violation-free of $<_{\mathrm{p}}$ for decompositions with respect to C are the words not containing ab, that is, all the words in $\Sigma^+ \setminus \Sigma^* ab \Sigma^* = a^+ \cup b^+ a^*$. The set of words which are violation-free of $<_{\mathrm{p}}$ for decodings with respect to C are the words in $\Sigma^+ \setminus C^* ab C^*$.

2. For the outfix relation, consider the language $C = \{aa, aba\}$ which is not an outfix code. A violation-free word for decompositions must not contain aba as an infix, that is, must be in $\Sigma^+ \setminus \Sigma^* aba \Sigma^*$. For decodings, such a word must not have the form $C^* aba C^*$.

3. For the intercode relation of index n, consider, without loss of generality, $n = 1$ and the language $C = \{ab, bba\}$. The language C is not an intercode of index 1, that is, not a comma-free code, as $\Sigma^+ C \Sigma^+ \cap C^2 \neq \emptyset$ with ab and $bbabba$ as witnesses. Note that C is a bifix code. For decompositions, $bbabba$ must not occur as an infix. For decodings, any word not in $C^* bbabba C^*$ is violation-free.

4. For the solid code relation, the infix part is analogous to $<_{\mathrm{p}}$ that has already been illustrated. The "new" problem is that of overlaps. Consider $C = \{ab, ba\}$, which is an infix code, but not an overlap-free language[10]. We focus on the overlap relation either in the form ω_{ol} or the form of $\overline{\omega_{\mathrm{ol}}}$. For decompositions the words which do not contain aba or bab are violation-free. For decodings, any word not in $C^* \{abab, baba\} C^*$ is violation-free.

Note that every non-empty word $q \notin C^+$ is violation-free for decodings with respect to C.

In general there is a pattern: For decompositions $q \notin \Sigma^* \mathsf{word}(\mathfrak{v}) \Sigma^*$ is violation-free. For decodings $q \notin C^* \mathsf{word}(\mathfrak{v}) C^*$ is violation-free.

When two relations interact, as in the case of solid codes, for violation-freeness the corresponding property seems not to be just a simple Boolean junction of the basic properties; this seems to require an expression of the co-locality of the respective defining situations. Neither Definition 3.4 based on Head's work nor our Definition 3.6 covers this adequately. We hope to look at this issue in a subsequent study.

Instead of violating instances one can also consider occurrences of words which, taken together, violate the condition in question although their occurrences may be "unrelated". To this end we modify Definition 3.1 following the pattern of Definition 3.6. In contrast to the violating instances, we consider a property P_ω which is given by an k-ary relation $\omega \subseteq k\text{-}\mathsf{tuples}(\Sigma^+)$. For example: for prefix-freeness we

[10] Note that overlap-freeness alone does not imply unique decodability.

have $(u, v) \in \omega_{\mathrm{p}}^{\neq}$ if $u <_{\mathrm{p}} v$; for overlap-freeness we have $(u, v) \in \omega_{\mathrm{ol}}$ if there exist $w_1, w_1, w_3 \in \Sigma^+$ such that $u = w_1 w_2$ and $v = w_2 w_3$; and for the intercode property of index n, we have

$$(u_1, u_2, \ldots, u_n, v_1, v_2, \ldots, v_{n+1}) \in \omega_{\mathrm{inter}_n}$$

if there exist $x, y \in \Sigma^+$ such that $v_1 v_2 \cdots v_{n+1} = x u_1 u_2 \cdots u_n y$. As our definition covers the overlap relation and a word can have a non-trivial overlap with itself, like $(xyx, xyx) \in \omega_{\mathrm{ol}}$, we cannot assume that the relation ω is irreflexive – if it is binary – in general. On the other hand, all binary relations ω_ϱ with $\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}\}$ are irreflexive. In order to make the following definition as general as possible, we let ω be an arbitrary subset of $\mathsf{all\text{-}tuples}(\Sigma^+)$ rather than a k-ary relation.

Definition 3.7. *Let $C \subseteq \Sigma^+$, $q \in \Sigma^+$, and $\omega \subseteq \mathsf{all\text{-}tuples}(\Sigma^+)$. Let P_ω be the predicate defining ω.*

1. *The word q is said to be* P_ω-admissible for decompositions with respect to C, *if, for all $\mathfrak{u} = (u_1, u_2, \ldots, u_n) \in \omega \cap \mathsf{all\text{-}tuples}(C)$, there exists (at least) one index $1 \leq i \leq n$ such that $u_i \not\leq_{\mathrm{i}} q$.*

2. *The word q is said to be* P_ω-admissible for decodings with respect to C, *if, for all $\mathfrak{u} = (u_1, u_2, \ldots, u_n) \in \omega \cap \mathsf{all\text{-}tuples}(C)$, there exists (at least) one index $1 \leq i \leq n$ such that there are no C-decodings $q = q_1 u_i q_2$ with $q_1, q_2 \in C^*$.*

Remark 3.2. For ω_{o} and the relations defining the shuffle hierarchy except $<_{\mathrm{i}}$ the definition of admissibility differs significantly from that of violation-freeness. Consider $u, v \in C$ with $(u, v) \in \omega_{\mathrm{o}}^{\neq}$. The occurrence of v would be a violating instance. However, it is no obstacle to admissibility unless also the word u occurs. This statement holds true for all shuffle relations including $<_{\mathrm{h}}$, but excluding $<_{\mathrm{i}}$.

For intercodes $\omega_{\mathrm{inter}_n}$, as well as comma-free codes and overlap-freeness, a word q is violation-free if the words in $\mathfrak{u} \in \omega_{\mathrm{inter}_n}$ do not appear in a particular constellation in q as defined by the binary relation $\overline{\omega_{\mathrm{inter}_n}}$. In contrast, for admissibility each word in $\mathfrak{u} \in \omega_{\mathrm{inter}_n}$ is treated individually and can appear anywhere in q.

Example 3.4. Let $\Sigma = \{a, b\}$.

1. Consider the prefix order $<_{\mathrm{p}}$ and the language $C = \{a, ab\}$. The set of words which are admissible for decompositions with respect to C are the words not containing ab, that is, all the words in $a^+ \cup b^+ a^*$; in this case violation-freeness and admissibility coincide because a is an infix of ab. The set of words which are admissible for decodings with respect to C are the words in $\Sigma^+ \setminus (C^* abC^* \cap C^* aC^*) = \Sigma^+ \setminus C^+ \cup a^+ \cup (ab)^+$.

2. For the outfix relation, consider the language $C = \{aa, aba\}$ which is not an outfix code. An admissible word for decompositions must not contain both aba and aa as infixes, that is, must be in $\Sigma^+ \setminus (\Sigma^* aba\Sigma^* \cap \Sigma^* aa\Sigma^*)$. For decodings, such a word must be in $\Sigma^+ \setminus (C^* abaC^* \cap C^* aaC^*)$.

3. For the comma-free relation, consider the language $C = \{ab, bba\}$. The language C is not a comma-free code, as $\Sigma^+C\Sigma^+ \cap C^2 \neq \emptyset$ with ab and $bbabba$ as witnesses. Note that C is a bifix code. For decompositions, a word is admissible if not both, bba and ab, are infixes of this word. For decodings, any word not in $C^*bbaC^* \cap C^*abC^*$ is admissible.

4. For solid codes, consider $C = \{ab, ba\}$, which is an infix code, but not an overlap-free language. We focus on the overlap relation in the form ω_{ol} rather than $\overline{\omega_{\mathrm{ol}}}$. For decompositions the words which do not contain ab and ba are admissible, that is, all words in $a^+b^* \cup b^*a^+ \cup b^+$. For decodings, any word not in $C^*abC^* \cap C^*baC^*$ is admissible.

The following two theorems show how the different notions of admissibility and violation-freeness are related to each other. The set of relations considered can be divided into two sets with two essentially different behaviours. The first set contains only binary, asymmetric, irreflexive relations and its properties are stated in Theorem 3.1; Figure 1 illustrates the relationships. The remaining properties are covered by Theorem 3.2 below.

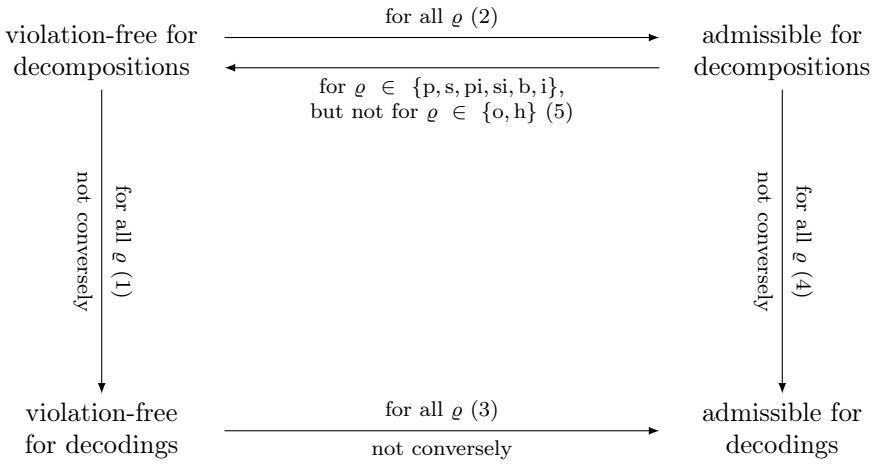

Figure 1: Relation described in Theorem 3.1: The numbers on the arrows refer to the statements in Theorem 3.1. This figure is restricted to $\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}\}$.

Theorem 3.1. *Let $C \subseteq \Sigma^+$, $q \in \Sigma^+$ and $\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}\}$. The following statements hold true:*

1. *If the word q is P_ϱ-violation-free for decompositions with respect to C, then it is P_ϱ-violation-free for decodings, but not conversely.*

2. *If the word q is P_ϱ-violation-free for decompositions with respect to C, then it is P_ϱ-admissible for decompositions with respect to C.*

3. *If the word q is P_ϱ-violation-free for decodings with respect to C, then it is P_ϱ-admissible for decodings with respect to C, but not conversely.*

4. *If the word q is P_ϱ-admissible for decompositions with respect to C, then it is P_ϱ-admissible for decodings with respect to C, but not conversely.*

5. *For $\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}\}$ the converse of (2) holds true. However, for $\varrho \in \{\mathrm{o}, \mathrm{h}\}$ the converse of (2) does not hold.*

Proof: Since $\omega_\varrho^{\neq}$ is binary and irreflexive, we consider two distinct words u and v such that $u\, \omega_\varrho^{\neq}\, v$. The words u and v are fixed throughout this proof.

Assume that the word q is P_ϱ-violation-free for decompositions with respect to C, that is, $v \not\leq_\mathrm{i} q$. In particular, v is not an infix of a decoding of q with respect to C.

Conversely, consider the language $C = \{ab, abab, aa, ba\}$ and note that $abab$ is the sole violating instance of P_ϱ for all relations considered here. The word $aababa$ with the unique C-decoding $(3, (aa, ba, ba))$ is P_ϱ-violation-free for decodings with respect to C, but it is not P_ϱ-violation-free for decompositions with respect to C. This proves (1).

Again, let q be P_ϱ-violation-free for decompositions with respect to C. As $v \not\leq_\mathrm{i} q$, trivially v and u cannot both be infixes of q. This proves (2).

Assume that the word q is P_ϱ-violation-free for decodings with respect to C. Then v is not an infix of a decoding of q with respect to C. Thus, trivially v and u cannot both be infixes of a decoding of q either.

Conversely, consider the language $C = \{ab, abbab\}$ and note that $ab\, \omega_\varrho^{\neq}\, abbab$ for all relations considered here. The word $abbab$ with the unique C-decoding $(1, (abbab))$ is not P_ϱ-violation-free for decodings with respect to C, but it is P_ϱ-admissible for decodings with respect to C. This proves (3).

Assume that the word q is P_ϱ-admissible for decompositions with respect to C. Then u and v are not both infixes of q. Hence, they are not both infixes of decodings of q with respect to C.

Conversely, consider the language $C = \{ab, abbab\}$ again, and note that $abbab$ is P_ϱ-admissible for decodings with respect to C, but it is not P_ϱ-admissible for decompositions with respect to C. This proves (4).

For $\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}\}$, if q is P_ϱ-admissible for decompositions with respect to C, then $v \not\leq_\mathrm{i} q$ because $u <_\mathrm{i} v$ due to the choice of ϱ. Hence, q is P_ϱ-violation-free for decompositions with respect to C.

Now, consider $C = \{aa, aba\}$, which is not an outfix code. The word aba contains aba, but not aa. Therefore, aba is P_o-admissible and P_h-admissible for decompositions with respect to C. On the other hand, the occurrence of aba is a P_o-violating and P_h-violating instance. This proves (5). □

The situation for overlap-free languages, solid codes, intercodes, and comma-free codes is different from the code properties that are covered by Theorem 3.1; Figure 2 illustrates the relationships stated in Theorem 3.2.

Theorem 3.2. *Let $C \subseteq \Sigma^+$, $q \in \Sigma^+$ and $\varrho \in \{\mathrm{ol}, \mathrm{solid}, \mathrm{inter}_n, \text{comma-free}\}$. The following statements hold true:*

1. *If the word q is P_ϱ-violation-free for decompositions with respect to C, then it is P_ϱ-violation-free for decodings, but not conversely.*
2. *If the word q is P_ϱ-admissible for decompositions with respect to C, then it is P_ϱ-violation-free for decompositions with respect to C, but not conversely.*
3. *If the word q is P_ϱ-admissible for decompositions with respect to C, then it is P_ϱ-admissible for decodings with respect to C, but not conversely.*
4. *If q is P_ϱ-asmissible for decodings with respect to C, this does not imply that q is P_ϱ-violation-free for decodings or decompositions with respect to C. If q is P_ϱ-violation-free for decodings or decompositions with respect to C, this does not imply that q is P_ϱ-admissible for decodings with respect to C.*
5. *If $q \in C^+$ and q is P_{solid}-violation-free for decodings with respect to C, then q is also P_{solid}-violation-free for decompositions with respect to C.*

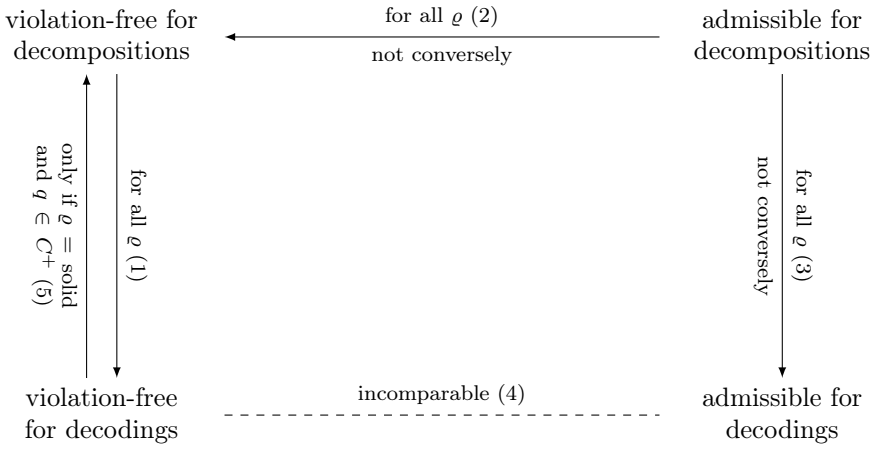

Figure 2: Relation described in Theorem 3.2: The numbers on the arrows refer to the statements in Theorem 3.2. This figure is restricted to $\varrho \in \{\mathrm{ol}, \mathrm{solid}, \mathrm{inter}_n, \text{comma-free}\}$.

Proof: Let $\varrho \in \{\mathrm{ol}, \mathrm{inter}_n\}$ and let $\mathfrak{u}, \mathfrak{v}$ be word tuples such that $\mathfrak{u}\ \overline{\omega_\varrho}\ \mathfrak{v}$. For $\varrho = \mathrm{inter}_n$, we require that $\mathfrak{u}, \mathfrak{v} \in \mathsf{all\text{-}tuples}(C)$ and we let $\mathfrak{w} = \mathfrak{u} \cdot \mathfrak{v}$ be the concatenation of the tuples $\mathfrak{u}$ and $\mathfrak{v}$. For $\varrho = \mathrm{ol}$, we only require that $\mathfrak{u} \in \mathsf{all\text{-}tuples}(C)$ and we let $\mathfrak{w} = \mathfrak{u}$. In both cases, we have $\mathfrak{w} \in \omega_\varrho \cap \mathsf{all\text{-}tuples}(C)$.

The case $P_{\text{comma-free}}$ is covered as a special case of P_{inter_n}, whereas the case $P_{\mathrm{solid}} = P_{\mathrm{i}} \wedge P_{\mathrm{ol}}$ requires special attention. Note that the positive statements of

(1,2,3) follow for P_{solid} because they hold for P_{ol} (proven below) and P_{i} (Theorem 3.1).

If q is P_ϱ-violation-free for decompositions with respect to C, then $\mathsf{word}(\mathfrak{v})$ is not a proper infix of q. In particular, $\mathsf{word}(\mathfrak{v})$ is not an infix of a decoding of q as described in Definition 3.6. Hence, q is P_ϱ-violation-free for decodings with respect to C. The same property holds for P_{solid} because it holds for P_{ol} and P_{i} by Theorem 3.1.

Conversely, for P_{inter_n} with $n \geq 1$, let $C = \{ab, ba, a(ab)^{n+1}\}$. The word $a(ab)^{n+1}$, with the unique C-decoding $(1, (a(ab)^{n+1}))$, is P_{inter_n}-violation-free for decodings with respect to C, but it is not P_{inter_n}-violation-free for decompositions with respect to C as it contains the violating instance $(ab)^{n+1}$. For P_{ol}, let $C = \{ab, ba, abaa\}$. The word $abaa$, with the unique C-decoding $(1, (abaa))$, is P_{ol}-violation-free for decodings with respect to C, but it is not P_{ol}-violation-free for decompositions with respect to C as it contains the violating instance aba. For P_{solid}, this result can only be obtained if q does not have a C-decoding and is, therefore, P_{solid}-violation-free for decodings with respect to C, but contains a P_{solid}-violating instance as infix; otherwise, statement (5) would be contradicted. This proves (1).

Assume q is P_ϱ-admissible for decompositions with respect to C. Hence, not all of the words in $\mathfrak{w}$ appear as infixes of q. Because $\mathsf{word}(\mathfrak{v})$ contains all words from $\mathfrak{w}$ as infixes, we have $\mathsf{word}(\mathfrak{v}) \not\leq_{\mathrm{i}} q$. Hence, q is P_ϱ-violation-free for decompositions with respect to C. This proves the "forward direction" of (2).

If q is P_ϱ-admissible for decompositions with respect to C, then not all of the words from $\mathfrak{w}$ can be infixes of q. In particular, they cannot all be infixes of decodings of q. Hence, q is P_ϱ-admissible for decodings with respect to C. This proves the "forward direction" of (3).

For $\varrho = \mathrm{inter}_n$ with $n \geq 1$, let $C = \{ab, ba\}$. The word $(ab)^{n+1}$ is P_{inter_n}-violation-free for decodings and decompositions with respect C, but it is P_{inter_n}-admissible for decodings with respect to C since ba does not appear in the unique C-decoding $(n+1, (ab, \ldots, ab))$. Furthermore, $(ab)^{n+1}$ is not P_{inter_n}-admissible for decompositions with respect to C. On the other hand, the word $abba$ with the unique decoding $(2, (ab, ba)$ is P_{inter_n}-violation-free for decodings and decompositions with respect to C, but it is not P_{inter_n}-admissible for decodings or decompositions with respect to C. Note that we do not require that ab or ba appears in n or $n+1$ distinct positions in $abba$. This proves (4) and the "converse directions" of (2) and (3) do not hold for intercodes of index n.

For $\varrho \in \{\mathrm{ol}, \mathrm{solid}\}$, let $C = \{abb, bab\}$. The word $abbabb$ is P_ϱ-admissible for decodings with respect to C, because bab does not occur in a decoding of $abbabb$. However, $abbabb$ contains the violating instance $abbab$ as described in Definition 3.6. Therefore, $abbabb$ is not P_ϱ-violation-free for decodings or decompositions with respect to C. Furthermore, $abbabb$ is not P_ϱ-admissible for decompositions with respect to C. The word $abbbab$, on the other hand, is P_ϱ-violation-free for decodings or decompositions with respect to C, but it is not P_ϱ-admissible for decodings or decompositions with respect to C. This proves (4) and the "converse directions" of (2) and (3) do not hold.

Let $q = u_1 u_2 \cdots u_n$ with $u_1, u_2, \ldots, u_n \in C$ be P_{solid}-violation-free for decodings

with respect to C. Suppose q contains a P_{solid}-violating instance v as infix If v it is a violating instance of P_{i}, let $w = v \in C$; if v it is a violating instance of P_{ol}, let $w <_{\text{p}} v$ such that $w \in C$ and there exists $w' <_{\text{s}} v$ such that $w' \in C$ and $|ww'| > |v|$. We distinguish five cases:

- If $w = u_i$, for some $1 \leq i \leq n$, such that $v \leq_{\text{p}} wu_{i+1} \cdots u_n$ then v would be a witness that q is not P_{solid}-violation-free for decodings with respect to C.
- If w was an infix of any u_i, then u_i would be a witness that q is not P_{i}-violation-free for decodings with respect to C.
- If $w = u_i u_{i+1} \cdots u_j$ for some i, j with $1 \leq i < j \leq n$, then w (in the decoding $q = u_1 \cdots u_{i-1} w u_{j+1} \cdots u_n$) would be a witness that q is not P_{i}-violation-free for decodings with respect to C.
- If there existed $1 \leq i < n$ and $x, y, z \in \Sigma^+$ such that $u_i = xy$, $w = yz$ and $w \leq_{\text{p}} u_{i+1} \cdots u_n$, then xyz is a witness that q is not P_{ol}-violation-free for decodings with respect to C.
- If there existed $1 < i \leq n$ and $x, y, z \in \Sigma^+$ such that $w = xy$, $u_i = yz$ and $w \leq_{\text{s}} u_1 \cdots u_{i-1}$, then xyz is a witness that q is not P_{ol}-violation-free for decodings with respect to C.

This covers all possibilities of how w, as prefix of v, can be located in q. This proves (5). □

One can intuit the relativization of a code property P as follows: If a word $q \in C^+$ satisfies the relativized property with respect to C, then the word should be uniquely decodable over C. As we show next, this intuition holds true for the notion of admissibility, but does so only for some special properties when considering violation-freeness. However, the converse of this statement is not true: If a word q is uniquely decodable over C, then q is not necessarily P-violation-free or P-admissible for decompositions or decodings with respect to C. For example, consider the prefix-free property and $C = \{ab, aba\}$. The word $q = ababa$ has the unique C-decoding $(2, (ab, aba))$; it is, however, neither P_{p}-violation-free nor P_{p}-admissible for decompositions or decodings with repect to C.

Theorem 3.3. *Let $\omega \subseteq k$-$\mathsf{tuples}(\Sigma^+)$ be a k-ary relation such that if a non-empty language D satisfies P_ω, then D is a code (all words over Σ^+ have at most one D-decoding). Let $C \subseteq \Sigma^+$ be a non-empty language. If q is P_ω-admissible for decodings or decompositions, then q has at most one C-decoding.*

Proof: Suppose $q \in \Sigma^+$ has two C-decodings and is P_ω-admissible for decodings or decompositions relative to C. Let

$$(m, (u_1, u_2, \ldots, u_m)) \text{ and } (n, (v_1, v_2, \ldots, v_n))$$

be two distinct C-decodings of q. Let $C' = \{u_1, u_2, \ldots, u_m, v_1, v_2, \ldots, v_n\} \subseteq C$. As q is P_ω-admissible for decodings or decompositions with respect to C, for all

$\mathfrak{w} \in k\text{-tuples}(C')$ we have $\mathfrak{w} \notin \omega$. Therefore, C' satisfies the property P_ω and must be a code. However, the word q has two C'-decodings – a contradiction! □

This result easily extends to violation-free words for the properties P_ϱ with $\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}, \mathrm{solid}\}$, using Theorem 3.1. For P_{solid} we need to observe that P_{solid}-violation-freeness with respect to a language C implies P_{i}-violation-freeness with respect to C.

Corollary 3.1. *Let $\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}, \mathrm{solid}\}$ and $C \subseteq \Sigma^+$ be a non-empty language. If q is P_ϱ-violation-free for decodings or decompositions, then q has at most one C-decoding.*

A similar result cannot be obtained for intercodes or comma-free codes: Let $C = \{ab, abab\}$ and $n \geq 1$ The word $abab$ clearly has two distinct C-decodings. However, $abab$ is P_{inter_n}-violation-free for decodings or decompositions with respect to C. Indeed, for comma-freeness the shortest violating instance over C is $(ab)\ \omega_{\text{comma-free}}\ (ab, abab)$ or $(ab)\ \omega_{\text{comma-free}}\ (abab, ab)$.

3.4 Relativized Codes

We have arrived at four notions of how a word may satisfy the predicate P_ϱ for a given non-empty language $C \subseteq \Sigma^+$:

1. vf-decomp: Violation-free for decompositions;
2. vf-decod: Violation-free for decodings;
3. adm-decomp: Admissible for decompositions;
4. adm-decod: Admissible for decodings.

Let $\mathfrak{M}$ be the set of these notions. Each $\mu \in \mathfrak{M}$ gives rise to a definition of a class of relativized codes as follows:

Definition 3.8. *Let C and L be non-empty subsets of Σ^+, let $\mu \in \mathfrak{M}$ and let*

$$\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}, \mathrm{solid}, \mathrm{ol}, \mathrm{inter}_n, \text{comma-free}\}.$$

The language C is a P_ϱ-μ code relative to L if every word in L has the property P_ϱ-μ with respect to C.

Let $\mathcal{C}^\mu_\varrho(L)$ be the class of P_ϱ-μ codes relative to L. Let $\mathcal{L}^\mu_\varrho(C)$ be the class of non-empty languages $L \subseteq \Sigma^+$ such that C is a P_ϱ-μ code relative to L. The following statements are consequences of the results obtained so far.

Theorem 3.4. *In the statements below the symbols μ, ϱ, C, and L are defined as follows: $\mu \in \mathfrak{M}$,*

$$\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}, \mathrm{solid}, \mathrm{ol}, \mathrm{inter}_n, \text{comma-free}\},$$

$C, L \subseteq \Sigma^+$, $C \neq \emptyset$, $L \neq \emptyset$. The following statements hold true:

1. *For all μ, ϱ and C, the set $\mathcal{L}^{\mu}_{\varrho}(C)$ is closed under arbitrary unions. Therefore, it contains a unique maximal language denoted $M^{\mu}_{C,\varrho}$.*
2. *For all μ, ϱ and L, the set $\mathcal{C}^{\mu}_{\varrho}(L)$ is closed under non-empty intersections.*
3. $\mathcal{C}^{\text{vf-decomp}}_{\varrho}(L) \subseteq \mathcal{C}^{\text{vf-decod}}_{\varrho}(L)$ *and* $\mathcal{C}^{\text{adm-decomp}}_{\varrho}(L) \subseteq \mathcal{C}^{\text{adm-decod}}_{\varrho}(L)$. *The inclusions are proper for some L.*
4. $\mathcal{C}^{\text{vf-decomp}}_{\varrho}(L) = \mathcal{C}^{\text{adm-decomp}}_{\varrho}(L)$ *for* $\varrho \in \{\text{p}, \text{s}, \text{pi}, \text{si}, \text{b}, \text{i}\}$; $\mathcal{C}^{\text{vf-decomp}}_{\varrho}(L) \subseteq \mathcal{C}^{\text{adm-decomp}}_{\varrho}(L)$ *for* $\varrho \in \{\text{o}, \text{h}\}$; *and* $\mathcal{C}^{\text{vf-decomp}}_{\varrho}(L) \supseteq \mathcal{C}^{\text{adm-decomp}}_{\varrho}(L)$ *for* $\varrho \in \{\text{solid}, \text{ol}, \text{inter}_n, \text{comma-free}\}$. *The inclusions are proper for some L.*
5. $\mathcal{C}^{\text{vf-decod}}_{\varrho}(L) \subseteq \mathcal{C}^{\text{adm-decod}}_{\varrho}(L)$ *for* $\varrho \in \{\text{p}, \text{s}, \text{pi}, \text{si}, \text{b}, \text{i}, \text{o}, \text{h}\}$. *The inclusion is proper for some L.*
6. *If $C' \subseteq C$, $C' \neq \emptyset$, then $\mathcal{L}^{\mu}_{\varrho}(C) \subseteq \mathcal{L}^{\mu}_{\varrho}(C')$ for all μ and ϱ.*
7. *If $L' \subseteq L$ then $\mathcal{C}^{\mu}_{\varrho}(L) \subseteq \mathcal{C}^{\mu}_{\varrho}(L')$ for all μ and ϱ.*

Proof: All statements are easy consequences of the definitions and of Theorems 3.1 and 3.2. □

Theorem 3.4 summarizes simple aspects of relativizing code properties. More detailed issues can be learned from Theorems 3.1 and 3.2. In both cases, the statements are limited to specific code properties ϱ. To identify the common scheme for a wider class of code properties is still an open problem.

3.5 The Old and New Definitions Compared

We outline how the definitions of code relativization given in [2] and [9] compare to the ones in the present paper. While we attempted to maintain consistency, it was inevitable that some definitions would change given the fact that a detailed look prompted by [13] revealed the need for a more general and less uniform approach. Hence, when reading the older papers together with this one, it is important to watch for slight, but possibly important, differences in the definitions, before using the statements of theorems. Particular attention needs to be paid to the issues in the following remark.

Remark 3.3. Let $L, C \subseteq \Sigma^+$ be non-empty languages.

1. Let P be a predicate on $\mathfrak{P}_{\leq 2}(C)$. Note that a subset ω of 1-$\mathsf{tuples}(\Sigma^+) \cup$ 2-$\mathsf{tuples}(\Sigma^+)$ describes the predicate P, that is $P = P_\omega$, if and only if

$$\begin{aligned} P(\{x,y\}) = 0 \quad &\Longleftrightarrow \quad (x,y) \in \omega \text{ or } (y,x) \in \omega \text{ and} \\ P(\{x\}) = 0 \quad &\Longleftrightarrow \quad (x) \in \omega \text{ or } (x,x) \in \omega. \end{aligned}$$

   Let $P = P_\omega$ be described by a set of tuples ω. A word $q \in C^+$ is P-admissible for C in the sense of Definition 3.1 if and only if it is P-admissible for decodings

with respect to C in the sense of Definition 3.7. Furthermore, if $L \subseteq C^+$, then C is a P-code relative to L in the sense of Definition 3.3 if and only if C is a P-admissible code for decodings relative to L in the sense of Definition 3.8.

Note that, Definitions 3.1 and 3.3 do not cover the cases when $q \notin C^+$ and $L \not\subseteq C^+$, respectively, while, in this paper, we naturally extend these definitions to all words $q \in \Sigma^+$ and languages $L \subseteq \Sigma^+$.

2. C is a solid code relative to L according to Definition 3.4 if and only if every word $q \in L$ is P_{solid}-violation-free for decompositions with respect to C in the sense of Definition 3.6 or, equivalently, if C is a P_{solid}-violation-free code for decompositions relative to L in the sense of Definition 3.8.

We suggest that the framework of this paper supersede those of [2, 9]. The concepts are still not ideal, but approaching what we consider the right ones.

4 Decidability Questions

In general, given non-empty languages $C, L \subseteq \Sigma^+$ and a code property P, one would like to know whether C is a P-code relative to L. In practical terms, we are given a language L of messages to be transmitted. We are also given a target for the transmission quality expressed by the predicate P. We want to know whether a given candidate code C serves the purpose. This gives rise to decidability problems for relativized codes.

For unrelativized codes, that is codes relative to Σ^+, results regarding the decidability of code properties as of 1996 are proved or summarized in [15, 14]. Further details are found in [3] and [4]. It is known that code properties are usually decidable when C is a regular language, and undecidable when C is a linear language. In [4] it is shown that the boundary between decidability and undecidability is significantly lower than that of linear languages. For relativized code properties this implies that one should not expect decidability unless C is regular. Regarding assumptions about L, we only consider the case of L being regular as well. At present we do not know to which extent this restriction can be lifted.

We first review two notions which we use in some of the proofs in this section.

1. Let $L \subseteq \Sigma^+$. The *syntactic congruence* $\sim_L$ with respect to L is defined as follows: For $u, v \in \Sigma^+$, $u \sim_L v$ if and only if, for all $x, y \in \Sigma^*$, either xuy and xvy are both in L or both not in L. The *syntactic semigroup* of L is the quotient semigroup $\Sigma^+/\sim_L$. Each element of the syntactic semigroup of L is a *syntactic class* which can be viewed as a language itself. For a word u we write $[u]_L$ to denote its syntactic class. The syntactic semigroup of a language L is finite if and only if L is regular. For languages L_1 and L_2 over the same alphabet Σ, $\sim_{(L_1,L_2)}$ denotes the intersection of the congruences $\sim_{L_1}$ and $\sim_{L_2}$. For additional basic information on syntactic semigroups we refer to [17, 24].

2. The second notion to consider is that of *shuffling on a trajectory*. This concept is widely used in order to describe code properties [29, 18, 19, 20, 21, 14]. A *trajectory* t is a word over the alphabet $\{0,1\}$. The result of shuffling two words u and v on the trajectory t is a word $w = u \sqcup\!\sqcup_t v$ that is obtained by using all letters from u and v where the trajectory t determines in which places to use letters from u or v. Shuffling is defined recursively by

$$\lambda \sqcup\!\sqcup_\lambda \lambda = \lambda, \quad au \sqcup\!\sqcup_{0t} v = a(u \sqcup\!\sqcup_t v), \quad u \sqcup\!\sqcup_{1t} bv = b(u \sqcup\!\sqcup_t v)$$

where $a, b \in \Sigma$, $u, v \in \Sigma^*$, and $t \in \{0,1\}^*$. Note that $u \sqcup\!\sqcup_t v$ is only defined if $|u| = |t|_0$ and $|v| = |t|_1$. This concept is extended to languages L_1, L_2 and a set of trajectories $\mathbf{t}$ by

$$L_1 \sqcup\!\sqcup_{\mathbf{t}} L_2 = \{u \sqcup\!\sqcup_t v \mid u \in L_1, v \in L_2, t \in \mathbf{t}\}.$$

The shuffle of two regular languages on a regular set of trajectories yields a regular language.

Let P_ϱ be a code property and $C \subseteq \Sigma^+$ be a non-empty language. For each of the four notions of relativized codes $\mu \in \mathfrak{M}$ there is a maximal language $M^{\mu}_{C,\varrho}$ such that a language L is a P_ϱ-μ code relative to C if and only if $L \subseteq M^{\mu}_{C,\varrho}$, as stated in Theorem 3.4.

We show that $M^{\text{vf-decomp}}_{C,\varrho}$ and $M^{\text{vf-decod}}_{C,\varrho}$ are effectively constructible regular languages for all P_ϱ considered in this paper. Thus, to decide whether or not a given regular language is a P_ϱ-violation-free code for decompositions or decodings relative to another regular language, one can test for inclusion of regular languages. Let $V_{C,\varrho} = \{v \in C \mid v \text{ is a violating instance of } P_\varrho \text{ in } C\}$. Here, a *violating instance* is a $\mathsf{word}(\mathfrak{v})$ as used in Definition 3.6. Since P_{ol}-violation-freeness and P_{solid}-violation-freeness, do not follow the general definition, these properties are treated separately.

Lemma 4.1. *Let $C \subseteq \Sigma^+$ be a non-empty language and let*

$$\varrho \in \{\text{p}, \text{s}, \text{pi}, \text{si}, \text{b}, \text{i}, \text{o}, \text{h}, \text{inter}_n, \text{comma-free}\}.$$

We have $M^{\text{vf-decomp}}_{C,\varrho} = \Sigma^+ \setminus \Sigma^* V_{C,\varrho} \Sigma^*$ and $M^{\text{vf-decod}}_{C,\varrho} = \Sigma^+ \setminus C^* V_{C,\varrho} C^*$.

Proof: The languages $M^{\text{vf-decomp}}_{C,\varrho}$ and $M^{\text{vf-decod}}_{C,\varrho}$ contain precisely those words which are violation-free for decompositions or decodings, respectively, with respect to C. This is a direct consequence of Definition 3.6. □

Theorem 4.1. *For*

$$\varrho \in \{\text{p}, \text{s}, \text{pi}, \text{si}, \text{b}, \text{i}, \text{o}, \text{h}, \text{ol}, \text{solid}, \text{inter}_n, \text{comma-free}\}$$

and regular $C \subseteq \Sigma^+$ the languages $M^{\text{vf-decomp}}_{C,\varrho}$ and $M^{\text{vf-decod}}_{C,\varrho}$ are effectively regular.

Proof: The sets of violations can be expressed as $V_{C,\mathrm{p}} = C \cap C\Sigma^+$, $V_{C,\mathrm{s}} = C \cap \Sigma^+ C$, $V_{C,\mathrm{pi}} = C \cap \Sigma^* C \Sigma^+$, $V_{C,\mathrm{si}} = C \cap \Sigma^+ C \Sigma^*$, $V_{C,\mathrm{b}} = V_{C,\mathrm{p}} \cup V_{C,\mathrm{s}}$, $V_{C,\mathrm{i}} = C \cap (\Sigma^+ C \Sigma^* \cup C\Sigma^+)$, $V_{C,\mathrm{inter}_n} = C^{n+1} \cap \Sigma^+ C^n \Sigma^+$, $V_{C,\text{comma-free}} = CC \cap \Sigma^+ C \Sigma^+$, which are all regular languages. The sets of violations for outfix-codes and hypercodes can be expressed using shuffling on a trajectory: we have $V_{C,\mathrm{h}} = C \cap (C \sqcup\!\sqcup_{\mathbf{t}_\mathrm{h}} \Sigma^+)$ for $\mathbf{t}_\mathrm{h} = \{0,1\}^+$ and $V_{C,\mathrm{o}} = C \cap (C \sqcup\!\sqcup_{\mathbf{t}_\mathrm{o}} \Sigma^+)$ for $\mathbf{t}_\mathrm{o} = 0^*1^+0^*$; both sets of violations are regular.

Using Lemma 4.1, we obtain that $M_{C,\varrho}^{\text{vf-decomp}}$ and $M_{C,\varrho}^{\text{vf-decod}}$ are regular because $V_{C,\varrho}$ is regular for $\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}, \mathrm{inter}_n, \text{comma-free}\}$. All steps in these constructions are effective.

For $\varrho = \mathrm{ol}$, the set of violations can be written as

$$V_{C,\mathrm{ol}} = \{xyz \mid x,y,z \in \Sigma^+, xy, yz \in C\} = \bigcup_{\substack{X,Y,Z \in \Sigma^+/\sim_C \\ XY \subseteq C, YZ \subseteq C}} XYZ$$

which is an effectively regular language. As before, we obtain $M_{C,\mathrm{ol}}^{\text{vf-decomp}} = \Sigma^+ \setminus \Sigma^* V_{C,\mathrm{ol}} \Sigma^*$.

The set $M_{C,\mathrm{ol}}^{\text{vf-decod}}$ cannot be expressed in a similar manner as above; nevertheless, it is effectively regular, given as

$$M_{C,\mathrm{ol}}^{\text{vf-decod}} = \Sigma^+ \setminus \bigcup_{\substack{X,Y,Z \in \Sigma^+/\sim_C \\ XY \subseteq C, YZ \subseteq C}} \left(C^* XY (Z\Sigma^* \cap C^+) \cup (C^+ \cap \Sigma^* X) YZ C^*\right).$$

Finally, we have $M_{C,\mathrm{solid}}^{\text{vf-decomp}} = M_{C,\mathrm{i}}^{\text{vf-decomp}} \cap M_{C,\mathrm{ol}}^{\text{vf-decomp}}$ and $M_{C,\mathrm{solid}}^{\text{vf-decod}} = M_{C,\mathrm{i}}^{\text{vf-decod}} \cap M_{C,\mathrm{ol}}^{\text{vf-decod}}$. □

Since the constructions of the regular languages in both Lemma 4.1 and Theorem 4.1 are effective, one concludes:

Corollary 4.1. *Let*

$$\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}, \mathrm{ol}, \mathrm{solid}, \mathrm{inter}_n, \text{comma-free}\}.$$

For given regular languages C and L it is decidable

1. *whether or not C is a P_ϱ-violation-free code for decompositions relative to L, and*
2. *whether or not C is a P_ϱ-violation-free code for decodings relative to L.*

This result can be extended to P_ϱ-admissible codes for decompositions for those properties for which P_ϱ-admissibility and P_ϱ-violation-freeness coincide – see Theorem 3.1.

Corollary 4.2. *Let*

$$\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}\}.$$

For given regular languages C and L it is decidable whether or not C is an P_ϱ-admissible code for decompositions relative to L.

The situation changes when considering admissibility for decodings. Decidability cannot be expressed as an inclusion test of two regular languages as before.

Proposition 4.1. *For a given regular $C \subseteq \Sigma^*$ the language $M_{C,p}^{\text{adm-decod}}$ is not necessarily regular.*

Proof: Let $\Sigma = \{0, 1\}$ and $C = 10^*$. Obviously, a word $w \in 10^i 10^j \in C^2$ belongs to $M_{C,\varrho}^{\text{adm-decod}}$ if and only if $i = j$. Therefore, the language $M_{C,\varrho}^{\text{adm-decod}}$ cannot be regular as its intersection with the regular language C^2 is not regular. □

Deciding whether or not a regular language C is an P_ϱ-admissible code for decodings relative to a regular language L works in two stages: first, decide whether or not C is a code relative to L, that is, every word in L has at most one C-decoding; then, verify that every decoding of a word in L is P_ϱ-admissible. We focus only on code properties P_ϱ defined by irreflexive binary relations; this excludes solid codes, intercodes, comma-free codes and overlap-free languages.

The next lemma forms the basis for deciding whether or not a regular language C is a code relative to a regular language L. The lemma itself does not require that the languages L and C be regular.

Lemma 4.2. *Let $L, C \subseteq \Sigma^+$ be non-empty languages. The language C is a code relative to L if and only if, for all syntactic classes $Y \in \Sigma^+/\sim_C$ contained in C, one has*

$$(C^*Y)^{-1}L \cap C^* \cap (Y^{-1}C \setminus \{\lambda\})C^* = \emptyset.$$

Proof: Suppose C is not a code relative to L. There exists a word $w = u_1 \cdots u_n = v_1 \cdots v_m \in L$ such that $u_1, \ldots, u_n, v_1, \ldots, v_m \in C$ and $u_i \neq v_i$ for some $1 \leq i \leq \min\{n, m\}$. Let i be minimal such that $u_i \neq v_i$ and, by symmetry, assume that $u_i <_p v_i$. Let $Y = [u_i]_C$ and observe that $z = u_{i+1} \cdots u_n$ belongs to $(C^*Y)^{-1}L$ as well as C^*; furthermore, since $u_i^{-1}v_i \in Y^{-1}C \setminus \{\lambda\}$, we obtain $z = (u_i^{-1}v_i)v_{i+1} \cdots v_m \in (Y^{-1}C \setminus \{\lambda\})C^*$. Therefore,

$$(C^*Y)^{-1}L \cap C^* \cap (Y^{-1}C \setminus \{\lambda\})C^*$$

is not empty.

Conversely, suppose that

$$z \in (C^*Y)^{-1}L \cap C^* \cap (Y^{-1}C \setminus \{\lambda\})C^*$$

exists for some $Y \in \Sigma^+/\sim_C$ with $Y \subseteq C$. Let $x_1, \ldots, x_i \in C^*$ and $y \in Y$ such that $w = x_1 \cdots x_i y z \in L$. There are $u_1, \ldots, u_n \in C$ such that $z = u_1 \cdots u_n$. Furthermore,

we let $v_0 \in y(Y^{-1}C \setminus \{\lambda\})$ and $v_1, \ldots, v_m \in C$ such that $yz = v_0, \ldots, v_m$; note that $v_0 \in C$. Thus, we found two factorizations

$$w = x_1 \cdots x_i y u_1 \cdots u_n = x_1 \cdots x_i v_0 \cdots v_m$$

of a word that belongs to L where all factors belong to C and $y \neq v_0$. We conclude that C is not a code relative to L. □

Theorem 3.3 and Lemma 4.2 lead to a general method for deciding whether a regular language C is an P_ϱ-admissible code for decodings relative to a regular language L provided the relation ω_ϱ is binary and recognizable in a transducer model with a decidable emptiness or membership problem. This applies to

$$\varrho \in \{\mathrm{p}, \mathrm{s}, \mathrm{pi}, \mathrm{si}, \mathrm{b}, \mathrm{i}, \mathrm{o}, \mathrm{h}\}.$$

In [15] it is shown that the emptiness problem for a transducer model is decidable if and only if its membership problem is decidable. Furthermore, if the emptiness problem of a transducer machine recognizing ω_ϱ is decidable, then for regular X, Y it is decidable whether or not $x \in X$ and $y \in Y$ exists such that $x \,\omega_\varrho\, y$.

Theorem 4.2. *Let $C, L \subseteq \Sigma^+$ be non-empty regular languages and let P_ϱ be a code property such that ω_ϱ is irreflexive and recognizable by a transducer machine with decidable emptiness problem. It is decidable whether or not C is a P_ϱ-admissible code for decodings relative to L.*

Proof: According to Theorem 3.3, for C to be an P_ϱ-admissible code for decodings relative to L, it is necessary that C is a code relative to L. By Lemma 4.2, we can decide whether or not C is a code relative to L by performing a series of emptiness test of regular languages. Henceforth, we assume that we have preformed this test and that C is a code relative to L.

We will show that, under the premise that C is a code relative to L, C is a P_ϱ-admissible code for decodings relative to L if and only if for all syntactic classes $X, Y \in \Sigma^+/{\sim_{(C,L)}}$ such that $X, Y \subseteq C$ and there exist $x \in X$ and $y \in Y$ with $x \,\omega_\varrho\, y$ or $y \,\omega_\varrho\, x$, we have

$$C^*XC^*YC^* \cap L = \emptyset.$$

Recall that one can decide whether or not there are $x \in X$ and $y \in Y$ such that $x \,\omega_\varrho\, y$ or $y \,\omega_\varrho\, x$ because ω_ϱ is recognizable by a transducer machine with decidable emptiness problem [15].

Now, suppose that $w \in C^*XC^*YC^* \cap L$ exists for a pair $X, Y \in \Sigma^+/{\sim_{(C,L)}}$ such that $X, Y \subseteq C$ and there exist $x \in X$ and $y \in Y$ with $x \,\omega_\varrho\, y$ or $y \,\omega_\varrho\, x$. Let $w \in u_1 X u_2 Y u_3$ for $u_1, u_2, u_3 \in C^*$. One obtains that $u_1 X u_2 Y u_3 \subseteq L$ and, therefore, $u_1 x u_2 y u_3 \in C^*XC^*YC^* \cap L$ with $x \,\omega_\varrho\, y$ or $y \,\omega_\varrho\, x$. Hence C is not a P_ϱ-admissible code for decodings relative to L.

Conversely, let $w \in L$ be a witness for the fact that C is not a P_ϱ-admissible code for decodings relative to L; that is, two words $x, y \in C$ such that $x \,\omega_\varrho\, y$ or $y \,\omega_\varrho\, x$ appear in decodings of w over C; note that we cannot have $x = y$ since ω_ϱ is

irreflexive. As C is a code relative to L, x and y appear in the same decoding; thus, without loss of generality, we can factorize $w = u_1xu_2yu_3$ with $u_1, u_2, u_3 \in C^*$ and $C^*[x]_{(C,L)}C^*[y]_{(C,L)}C^* \cap L$ cannot be empty. □

With the tools used in this section we cannot answer the following questions:

1. For $\varrho \in \{\mathrm{o}, \mathrm{h}, \mathrm{ol}, \mathrm{solid}, \mathrm{inter}_n, \text{comma-free}\}$ and given regular languages C, L, is it decidable whether or not C is a P_ϱ-admissible code for decompositions relative to L?

2. For $\varrho \in \{\mathrm{ol}, \mathrm{solid}, \mathrm{inter}_n, \text{comma-free}\}$ and given regular languages C, L, is it decidable whether or not C is a P_ϱ-admissible code for decodings relative to L?

5 Final Remarks

In information processing, coding serves several purposes. These are expressed by code properties. In a real information transmission system, messages arrive with different probabilities including many with probability 0. Unless data ideal compression is applied, which would essentially eliminate the latter and make all messages equally likely, coding should take into account which messages are likely to be encoded. This idea is modelled by relativized codes. Thus the standard code properties, both information theoretically and in terms of combinatorics, have their relativized counterparts, relativized to the language of likely messages to be encoded. This is very much in the spirit of Shannon's channel coding theorem [26] where messages of probability 0 are practically ignored.

Contrary to what was envisaged in [2], no uniformly acceptable relativization seems possible. Instead, examining various potential models, we arrived at four definitions, each of which seems to be equally well motivated.

We compare these models both among each other and to intuitive expectations. We also consider their decidability properties. We have indicated a few open questions. Many more could have been mentioned.

Acknowledgement: Research reported in this paper was supported by grants from the Natural Sciences and Engineering Council of Canada.

References

[1] J. Berstel, D. Perrin, C. Reutenauer: *Codes and Automata. Encyclopedia of Mathematics and Its Applications* **129**. Cambridge University Press, Cambridge, 2010, xiv + 620 pp.

[2] M. Daley, H. Jürgensen, L. Kari, K. Mahalingam: Relativized codes. *Theoret. Comput. Sci.* **429** (2012), 54–64.

[3] K. Dudzinski, S. Konstantinidis: Formal descriptions of code properties: Decidability, complexity, implementation. *Internat. J. Foundations Comput. Sci.* **23**(1) (2012), 67–85.

[4] H. Fernau, K. Reinhardt, L. Staiger: Decidability of code properties. *RAIRO Inform. Théor. Appl.* **41**(3) (2007), 243–259.

[5] E. R. Gümüştop: *Varieties of Codes and Codifiable Varieties of Monoids*. PhD Dissertation, Binghampton University, State University of New York, 1997.

[6] F. Guzmán: Decipherability of codes. *J. Pure Appl. Algebra* **141** (1999), 13–35.

[7] T. Harju, J. Karhumäki: On the defect theorem and simplifiability. *Semigroup Forum* **33** (1986), 199–217.

[8] T. Head, A. Weber: Deciding multiset decipherability. *IEEE Trans. Inform. Theory* **41** (1995), 291–297.

[9] T. Head: Unique decipherability relative to a language. *Tamkang J. Math.* **11** (1980), 59–66.

[10] T. Head: Deciding the immutability of regular codes and languages und finite transductions. *Inform. Process. Lett.* **31** (1989), 239–241.

[11] T. Head: Relativized code concepts and multi-tube DNA dictionaries. In C. S. Calude, G. Păun (editors): *Finite versus Infinite – Contributions to an Eternal Dilemma*. 175–186. Springer-Verlag, London, 2000.

[12] T. Head. Draft of notes for Form. Lang. Sem. Thurs. Sept. 12, 2002, 2002, 3 pp. Personal Communication.

[13] H. Jürgensen: Markers and deterministic acceptors for non-deterministic languages. *J. of Automata, Languages and Combinatorics* **14** (2009), 33–62. Special issue on the occasion of D. Wotschke's sixtieth birthday.

[14] H. Jürgensen, S. Konstantinidis: Codes. In Rozenberg and Salomaa [25], 1, 511–607.

[15] H. Jürgensen, K. Salomaa, S. Yu: Transducers and the decidability of independence in free monoids. *Theoret. Comput. Sci.* **134** (1994), 107–117.

[16] H. Jürgensen, S. S. Yu: Relations on free monoids, their independent sets, and codes. *Internat. J. Comput. Math.* **40** (1991), 17–46.

[17] G. Lallement: *Semigroups and Combinatorial Applications*. John Wiley & Sons, New York, 1979.

[18] D. Y. Long: k-Outfix codes. *Chinese Ann. Math. Ser. A* **10** (1989), 94–99, in Chinese.

[19] D. Y. Long: k-Prefix codes and k-infix codes. *Acta Math. Sinica* **33** (1990), 414–421, in Chinese.

[20] D. Y. Long: On the structure of some group codes. *Semigroup Forum* **45** (1992), 38–44.

[21] D. Y. Long: k-Bifix codes. *Riv. Mat. Pura Appl.* **15** (1994), 33–55.

[22] D. J. C. MacKay: *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, Cambridge, 6th ed., 2007.

[23] K. Mahalingam: *Involution Codes: with Application to DNA Strand Design*. PhD thesis, University of South Florida, 2004, (3)+ii+70 pp.

[24] J.-E. Pin: Syntactic semigroups. In Rozenberg and Salomaa [25], 1, 679–746.

[25] G. Rozenberg, A. Salomaa (editors): *Handbook of Formal Languages*. Springer-Verlag, Berlin, 1997.

[26] C. E. Shannon: A mathematical theory of communication. *Bell System Tech. J.* **27** (1948), 379–423, 623–656.

[27] H. J. Shyr: *Free Monoids and Languages*. Hon Min Book Company, Taichung, third ed., 2001, iv+282 pp.

[28] H. J. Shyr, G. Thierrin: Codes and binary relations. In M. P. Malliavin (editor): *Séminaire d'algèbre Paul Dubreil, Paris 1975–1976, (29ème Année)*. *Lecture Notes in Computer Science* **586**, 180–188, Springer-Verlag, Berlin, 1977.

[29] G. Thierrin, S. S. Yu: Shuffle relations and codes. *J. Inform. and Optim. Sci.* **12** (1991), 441–449.

[30] A. Weber, T. Head: The finest homophonic partition and related code concepts. *IEEE Trans. Inform. Theory* **42** (1996), 1569–1575.

[31] S.-S. Yu: *Languages and Codes*. Tsang Hai Book Publishing Co., Taichung, Taiwan, 2005.

Remembering Ferenc Gécseg

This paper is dedicated to Ferenc Gécseg. By his influential work he is known to and admired by, all authors of this paper.

The first author, "I" in the sequel, was a close friend and is adding a few personal memories.

Between 1970 and 2000 I was a frequent visitor to Hungary mostly working with Jenő Szép and István Peák in Budapest, but also with colleagues at the Mathematics Department in Szeged, in particular with Ferenc. As we grew familiar, we usually spent the time in his garden, a bit outside Szeged, and research ideas grew among grapes and fruits. We shared many interests including universal algebra and

automaton theory of course, but also gardening, and social politics – and even the birthday. We wrote several papers together: on dependence in algebras, on algebras with dimension, on soliton automata, on products of automata, and on automata over algebras.

Ferenc and his wife Maria visited us in London, Canada, in 1987/88 for nearly a year. This is when Ferenc taught me much of what I did not know in universal algebra. He also tried to teach me how to turn the soil in our garden; that was not quite as successful, because our soil is very heavy clay unlike the sand in the Szeged area. We visited each other and met on many occasions, including several important seminars in Bulgaria. Exchanging thoughts on all kinds of matters, including scientific, political and social issues, was always an exceptionally positive experience.

I was in contact with Ferenc about a week before his death. He did not sound confident but expressed some hope. I and my wife miss him and his wife Maria, who died a few years ago.

Received 23rd July 2015

Acta Cybernetica **22** (2015) 293–311.

Quotient Complexities of Atoms in Regular Ideal Languages*

Janusz Brzozowski† and Sylvie Davies‡

Abstract

A (left) quotient of a language L by a word w is the language $w^{-1}L = \{x \mid wx \in L\}$. The quotient complexity of a regular language L is the number of quotients of L; it is equal to the state complexity of L, which is the number of states in a minimal deterministic finite automaton accepting L. An atom of L is an equivalence class of the relation in which two words are equivalent if for each quotient, they either are both in the quotient or both not in it; hence it is a non-empty intersection of complemented and uncomplemented quotients of L. A right (respectively, left and two-sided) ideal is a language L over an alphabet Σ that satisfies $L = L\Sigma^*$ (respectively, $L = \Sigma^*L$ and $L = \Sigma^*L\Sigma^*$). We compute the maximal number of atoms and the maximal quotient complexities of atoms of right, left and two-sided regular ideals.

Keywords: atom, left ideal, quotient, quotient complexity, regular language, right ideal, state complexity, syntactic semigroup, two-sided ideal

I dedicate this work to the memory of Ferenc Gécseg. I have known Ferenc for many years not only as an eminent scientist, author of numerous publications, and editor of Acta Cybernetica, but also as a good friend. I fondly remember his generosity and hospitality during my frequent visits to Hungary.

Janusz Brzozowski

1 Introduction

We assume that the reader is familiar with basic concepts of regular languages and finite automata; more background is given in the next section. Consider a regular language L over a finite non-empty alphabet Σ. Let $\mathcal{D} = (Q, \Sigma, \delta, q_1, F)$ be a minimal *deterministic finite automaton (DFA)* recognizing L, where Q is the set of *states*, $\delta\colon Q \times \Sigma \to Q$ is the *transition function*, q_1 is the *initial* state, and $F \subseteq Q$

*This work was supported by the Natural Sciences and Engineering Research Council of Canada under grant No. OGP0000871.

†David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, ON, Canada N2L 3G1. E-mail: brzozo@uwaterloo.ca

‡Department of Pure Mathematics, University of Waterloo, Waterloo, ON, Canada N2L 3G1. E-mail: sldavies@uwaterloo.ca

is the set of *final* states. There are three natural equivalence relations associated with L and $\mathcal{D}$.

The *Nerode right congruence* [14] is defined as follows: Two words x and y are equivalent if for every $v \in \Sigma^*$, xv is in L if and only if yv is in L. The set of all words that "can follow" a given word x in L is the *left quotient of L by x*, defined by $x^{-1}L = \{v \mid xv \in L\}$. In automaton-theoretic terms $x^{-1}L$ is the set of all words v that are accepted from the state $q = \delta(q_1, x)$ reached when x is applied to the initial state of $\mathcal{D}$; this is known as the *right language* of state q, the language accepted by DFA $\mathcal{D}_q = (Q, \Sigma, \delta, q, F)$. The Nerode equivalence class containing x is known as the *left language* of state q, the language accepted by DFA ${}_q\mathcal{D} = (Q, \Sigma, \delta, q_1, \{q\})$. The number n of Nerode equivalence classes is the number of distinct left quotients of L, known as its *quotient complexity* [1]. This is the same number as the number of states in $\mathcal{D}$, and is therefore known as L's *state complexity* [16]. Quotient/state complexity is now a commonly used measure of complexity of a regular language, and constitutes a basic reference for other measures of complexity. One can also define the quotient complexity of a Nerode equivalence class, that is, of the language accepted by DFA ${}_q\mathcal{D}$. In the worst case – for example, if $\mathcal{D}$ is strongly connected – this is n for every q.

The *Myhill congruence* [13] refines the Nerode right congruence and is a (two-sided) congruence. Here a word x is equivalent to a word y if for all u and v in Σ^*, uxv is in L if and only if uyv is in L. This is also known as the *syntactic congruence* [15] of L. The quotient of the free semigroup Σ^+ by this congruence is the *syntactic semigroup* of L. In automaton-theoretic terms two words are equivalent if they induce the same transformation of the set of states of a minimal DFA of L. The quotient complexity of Myhill classes has not been studied.

The third equivalence, which we call the *atom congruence* is a left congruence refined by the Myhill congruence. Here two words x and y are equivalent if $ux \in L$ if and only if $uy \in L$ for all $u \in \Sigma^*$. Thus x and y are equivalent if $x \in u^{-1}L$ if and only if $y \in u^{-1}L$. An equivalence class of this relation is called an *atom* of L [9]. It follows that an atom is a non-empty intersection of complemented and uncomplemented quotients of L.

The atom congruence is related to the Myhill and Nerode congruences in a natural way. Say a congruence on Σ^* *recognizes* L if L can be written as a union of the congruence's classes. The Myhill congruence is the unique *coarsest* congruence (that is, the one with the fewest equivalence classes) that recognizes L [15]. The Nerode and atom congruences are respectively the coarsest *right* and *left* congruences that recognize L.

The quotient complexity of atoms of regular languages has been studied in [4, 8, 12]. In this paper we study the quotient complexity of atoms in three subclasses of regular languages, namely, right, left, and two-sided ideals.

Ideals are fundamental concepts in semigroup theory. A language L over an alphabet Σ is a *right* (respectively, *left* and *two-sided*) *ideal* if $L = L\Sigma^*$ (respectively, $L = \Sigma^*L$ and $L = \Sigma^*L\Sigma^*$). The quotient complexity of operations on regular ideal languages has been studied in [6], and the reader should refer to that paper for more information about ideals. Ideals appear in pattern matching. A right (left)

ideal $L\Sigma^*$ (Σ^*L) represents the set of all words beginning (ending) with some word of a given set L, and $\Sigma^*L\Sigma^*$ is the set of all words containing a factor from L.

2 Preliminaries

It is well known that a language $L \subseteq \Sigma^*$ is regular if and only if it has a finite number of quotients. We denote the number of quotients of L (the *quotient complexity*) by $\kappa(L)$. This is the same as the *state complexity*, the number of states in a minimal DFA of L. Since we will not be discussing other measures of complexity, we refer to both quotient and state complexity as just *complexity*.

Let the set of quotients of a regular language L be $K = \{K_1, \ldots, K_n\}$. The *quotient automaton* of L is the DFA $\mathcal{D} = (K, \Sigma, \delta, L, F)$, where $\delta(K_i, a) = K_j$ if $a^{-1}K_i = K_j$, $L = K_1 = \varepsilon^{-1}L$ by convention, and $F = \{K_i \mid \varepsilon \in K_i\}$. This DFA is uniquely defined by L and is isomorphic to every minimal DFA of L.

A *transformation* of a set Q_n of n elements is a mapping of Q_n *into* itself, whereas a *permutation* of Q_n is a mapping of Q_n *onto* itself. In this paper we consider only transformations of finite sets, and we assume without loss of generality that $Q_n = \{1, \ldots, n\}$. An arbitrary transformation has the form

$$t = \begin{pmatrix} 1 & 2 & \cdots & n-1 & n \\ i_1 & i_2 & \cdots & i_{n-1} & i_n \end{pmatrix},$$

where $i_k \in Q_n$ for $1 \leqslant k \leqslant n$. The image i_j of element j under transformation t is denoted by jt. The image of $S \subseteq Q_n$ is $St = \bigcup_{j \in S}\{jt\}$. The *identity* transformation $\mathbf{1}$ maps each element to itself. For $k \geqslant 2$, a transformation (permutation) t is a *k-cycle* if there is a set $P = \{q_1, q_2, \ldots, q_k\} \subseteq Q_n$ such that if $q_1t = q_2, q_2t = q_3, \ldots, q_{k-1}t = q_k, q_kt = q_1$, and $qt = q$ for all $q \notin P$. A k-cycle is denoted by $(q_1, q_2, \ldots, q_k)$. A 2-cycle (q_1, q_2) is called a *transposition*. A transformation is *constant* if it maps all states to a single state q; we denote it by $(Q_n \to q)$. A transformation t that maps p to q, $q \neq p$ and does not afffect any $r \neq p$ is denoted by $(p \to q)$. The set of all transformations of Q_n is a monoid under composition, called the *complete transformation monoid* and denoted by T_n. The following is well-known:

Proposition 1. *The complete transformation monoid T_n has size n^n and can be generated by $\{(1, \ldots, n), (1, 2), (n \to 1)\}$, and by $\{(1, \ldots, n), (2, \ldots, n), (n \to 1)\}$.*

For a DFA $\mathcal{D} = (Q, \Sigma, \delta, q_1, F)$ we define the transformations $\{\delta_w \mid w \in \Sigma^+\}$ by $q\delta_a = \delta(q, a)$ for $a \in \Sigma$, and $q\delta_w = q\delta_x\delta_a$ for $w = xa$, $x \in \Sigma^*$. This set is a semigroup under composition and it is called the *transition semigroup* of $\mathcal{D}$. We also define $\delta_\varepsilon = \mathbf{1}$. The transformation δ_w is called the *transformation induced by* w. To simplify notation, we usually make no distinction between the word $w \in \Sigma^*$ and the transformation δ_w. If $\mathcal{D}$ is the quotient automaton of L, then the transition semigroup of $\mathcal{D}$ is isomorphic to the syntactic semigroup of L [15]. A state $q \in Q$ is *reachable from* $p \in Q$ if $pw = q$ for some $w \in \Sigma^*$, and *reachable* if it is reachable from q_1. Two states p, q are *indistinguishable* if $pw \in F \Leftrightarrow qw \in F$ for all $w \in \Sigma^*$,

and *distinguishable* otherwise. Indistinguishability is an equivalence relation on Q; furthermore, if $\mathcal{D}$ recognizes a language L, we can compute $\kappa(L)$ by counting the number of equivalence classes under indistinguishability of the reachable states of $\mathcal{D}$. A state is *empty* if its right language (defined in the introduction) is $\emptyset$.

3 Atoms

Atoms of regular languages were studied in [9], and their complexities in [3, 8, 12]. As discussed earlier, atoms are the classes of the *atom congruence*, a left congruence which is the natural counterpart of the Myhill two-sided congruence and Nerode right congruence. The Myhill and Nerode congruences are fundamental in regular language theory, but it seems comparatively little attention has been paid to the atom congruence and its classes. In [2] it was argued that it is useful to consider the complexity of a language's atoms when searching for complex regular languages, since one would expect such languages to have complex atoms.

Below we present an alternative characterization of atoms, which we use in our proofs. Earlier papers on atoms such as [3, 8, 9] take this as the definition of atoms, for it was not known until recently that atoms may be viewed as congruence classes (this fact was first noticed by Iván in [12]).

From now on assume all languages are non-empty. Denote the complement of a language L by $\overline{L} = \Sigma^* \setminus L$. Let $Q_n = \{1, \ldots, n\}$ and let L be a regular language with quotients $K = \{K_1, \ldots, K_n\}$. Each subset S of Q_n defines an *atomic intersection* $A_S = \bigcap_{i \in S} K_i \cap \bigcap_{i \in \overline{S}} \overline{K_i}$, where $\overline{S} = Q_n \setminus S$. An *atom* of L is a non-empty atomic intersection. Since atoms are pairwise disjoint, every atom A has a unique atomic intersection associated with it, and this atomic intersection has a unique subset S of K associated with it. This set S is called the *basis* of A.

Throughout the paper, L is a regular language of complexity n with quotients $K_1, \ldots, K_n$ and minimal DFA $\mathcal{D} = (Q_n, \Sigma, \delta, 1, F)$ such that the language of state i is K_i. Let $A_S = \bigcap_{i \in S} K_i \cap \bigcap_{i \in \overline{S}} \overline{K_i}$ be an atom. For any $w \in \Sigma^*$ we have

$$w^{-1}A_S = \bigcap_{i \in S} w^{-1}K_i \cap \bigcap_{i \in \overline{S}} \overline{w^{-1}K_i}.$$

Since a quotient of a quotient of L is also a quotient of L, $w^{-1}A_S$ has the form;

$$w^{-1}A_S = \bigcap_{i \in X} K_i \cap \bigcap_{i \in Y} \overline{K_i},$$

where $|X| \leqslant |S|$ and $|Y| \leqslant n - |S|$, $X, Y \subseteq Q_n$.

The complexity of atoms of a regular language was computed in [8] using a unique NFA defined by L_n, called the *átomaton*. In that NFA the language of each state q_S is an atom A_S of L_n. To find the complexity of that atom, the átomaton started in state q_S was converted to an equivalent DFA. A more direct and simpler method was used by Szabolcs Iván [12] who constructed the DFA for the atom directly from the DFA $\mathcal{D}_n$. We follow that approach here and outline it briefly for completeness.

For any regular language L an atom A_S corresponds to the ordered pair $(S, \overline{S})$, where S ($\overline{S}$) is the set of subscripts of uncomplemented (complemented) quotients. If L is represented by a DFA $\mathcal{D} = (Q, \Sigma, \delta, q_1, F)$, it is more convenient to think of S and $\overline{S}$ as subsets of Q. Similarly, any quotient of A_S corresponds to a pair (X, Y) of subsets of Q. For the quotient of A_S reached when a letter $a \in \Sigma$ is applied to the quotient corresponding to (X, Y) we get

$$a^{-1}\left(\bigcap_{i\in X} K_i \cap \bigcap_{i\in Y} \overline{K_i}\right) = \bigcap_{i\in X} a^{-1}K_i \cap \bigcap_{i\in Y} \overline{a^{-1}K_i} = \bigcap_{i\in X} K_{ia} \cap \bigcap_{i\in Y} \overline{K_{ia}}.$$

In terms of pairs of subsets of Q, from (X, Y) we reach (Xa, Ya). Note that if $X \cap Y \neq \emptyset$ in (X, Y) then the corresponding quotient is empty. Note also that the quotient of atom A_S corresponding to (X, Y) is final if and only if each quotient K_i with $i \in X$ contains ε, and each K_j with $j \in Y$ does not contain ε.

These considerations lead to the following definition of a DFA for A_S [12]:

Definition 1. *Suppose $\mathcal{D} = (Q, \Sigma, \delta, q_1, F)$ is a DFA and let $S \subseteq Q$. Define the DFA $\mathcal{D}_S = (Q_S, \Sigma, \Delta, (S, \overline{S}), F_S)$, where*

- $Q_S = \{(X, Y) \mid X, Y \subseteq Q, X \cap Y = \emptyset\} \cup \{\bot\}$.
- *For all $a \in \Sigma$, $\Delta((X, Y), a) = (\delta(X, a), \delta(Y, a))$ if $\delta(X, a) \cap \delta(Y, a) \neq \emptyset$, and $\Delta((X, Y), a) = \bot$ otherwise; and $\Delta(\bot, a) = \bot$.*
- $F_S = \{(X, Y) \mid X \subseteq F, Y \subseteq \overline{F}\}$.

DFA $\mathcal{D}_S$ recognizes the atomic intersection A_S of L. If $\mathcal{D}_S$ recognizes a non-empty language, then A_S is an atom.

4 Complexity of Atoms in Regular Languages

Upper bounds on the maximal complexity of atoms of regular languages were derived in [8]; for completeness we include these results. For $n = 1$ there is only one non-empty language $L = \Sigma^*$; it has one atom, L, which is of complexity 1. From now on assume that $n \geqslant 2$.

Proposition 2. *Let L be a regular language with $n \geqslant 2$ quotients. Then L has at most 2^n atoms. If $S \in \{Q_n, \emptyset\}$, then $\kappa(A_S) \leqslant 2^n - 1$. Otherwise,*

$$\kappa(A_S) \leqslant 1 + \sum_{x=1}^{|S|} \sum_{y=1}^{n-|S|} \binom{n}{x}\binom{n-x}{y}.$$

Proof. Since the number of subsets S of Q_n is 2^n, there are at most that many atoms. For atom complexity consider the following three cases:

1. $S = Q_n$. Then $A_{Q_n} = \bigcap_{i\in Q_n} K_i$ is the intersection of all quotients of L. For $w \in \Sigma^*$, we have $w^{-1}A_{Q_n} = \bigcap_{i\in X} K_i$, where $1 \leqslant |X| \leqslant |Q_n|$. Hence there are at most $2^n - 1$ quotients of this atom.

2. $S = \emptyset$. Now $A_\emptyset = \bigcap_{i \in Q_n} \overline{K_i}$, and $w^{-1}A_\emptyset = \bigcap_{i \in Y} \overline{K_i}$, where $1 \leqslant |Y| \leqslant |Q_n|$. As in the first case, there are at most $2^n - 1$ quotients of this atom.

3. $\emptyset \subsetneq S \subsetneq Q_n$. Then $A_S = \bigcap_{i \in S} K_i \cap \bigcap_{i \in \overline{S}} \overline{K_i}$. Every quotient of A_S has the form $w^{-1}A_S = \bigcap_{i \in X} K_i \cap \bigcap_{i \in Y} \overline{K_i}$, where $1 \leqslant |X| \leqslant |S|$ and $1 \leqslant |Y| \leqslant n - |S|$. There are two subcases:

   a) If $X \cap Y \neq \emptyset$, then $w^{-1}A_S = \emptyset$.

   b) If $X \cap Y = \emptyset$, there are at most $\sum_{x=1}^{|S|} \sum_{y=1}^{n-|S|} \binom{n}{x}\binom{n-x}{y}$ quotients of A_S of this form. This follows since $\binom{n}{x}$ is the number of ways to choose a set $X \subseteq Q_n$ of size x, and once X is fixed, $\binom{n-x}{y}$ is the number of ways to choose a set $Y \subseteq Q_n$ of size y that is *disjoint* from X. Taking the sum over the permissible values of x and y gives the formula above.

   Adding the results of (a) and (b) we have the required bound.

□

It was shown in [2] that the language L_n accepted by the minimal DFA $\mathcal{D}_n$ of Definition 2, also illustrated in Figure 1, meets all the complexity bounds for common operations on regular languages.

Definition 2. *For $n \geqslant 2$, let $\mathcal{D}_n = (Q_n, \Sigma, \delta_n, 1, \{n\})$, where $Q_n = \{1, \ldots, n\}$ is the set of states, $\Sigma = \{a, b, c\}$ is the alphabet, the transition function δ_n is defined by $a = (1, \ldots, n)$, $b = (1, 2)$, and $c = (n \to 1)$, state 1 is the initial state, and $\{n\}$ is the set of final states. Let L_n be the language accepted by $\mathcal{D}_n$. (If $n = 2$, a and b induce the same transformation; hence $\Sigma = \{a, c\}$ suffices.)*

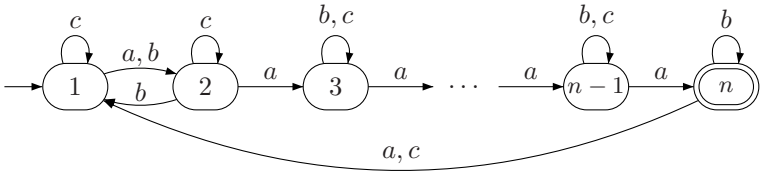

Figure 1: DFA of a regular language whose atoms meet the bounds.

It was proved in [8] that L_n has 2^n atoms, all of which are as complex as possible. We include the proof of this theorem following [12]. We first prove a general result about distinguishability of states in $\mathcal{D}_S$, which we will use throughout the paper.

Lemma 1 (Distinguishability)**.** *Let $\mathcal{D} = (Q, \Sigma, \delta, q_1, F)$ be a minimal DFA and for $S \subseteq Q$, let $\mathcal{D}_S = (Q_S, \Sigma, \Delta, (S, \overline{S}), F_S)$ be the DFA of the atom A_S. Then:*

1. *States (X, Y) and (X', Y') of $\mathcal{D}_S$ are distinguishable if $X \neq X'$ and $A_X, A_{X'}$ are both atoms, or if $Y \neq Y'$ and $A_{\overline{Y}}, A_{\overline{Y'}}$ are both atoms.*

2. *If one of A_X or $A_{\overline{Y}}$ is an atom, then (X, Y) is distinguishable from $\perp$.*

Proof. First note that if A_Z is an atom, then the initial state of $\mathcal{D}_Z$ must be non-empty, so there is a word w_Z in the transition semigroup of $\mathcal{D}$ such that $(Z, \overline{Z})w_Z = (U, V)$ with $U \subseteq F$, $V \subseteq \overline{F}$, i.e., $(U, V) \in F_S$. In particular, $(X, Y)w_X \in F_S$, since $Y \subseteq \overline{X}$. We also have $(X, Y)w_{\overline{Y}} \in F_S$, since Y is sent to a subset of $\overline{F}$, and $X \subseteq \overline{Y}$ is sent to a subset of F. This proves (2): if one of A_X or $A_{\overline{Y}}$ is an atom, then one of w_X or $w_{\overline{Y}}$ is in the transition semigroup of $\mathcal{D}$, and hence (X, Y) can be mapped to a final state but $\perp$ cannot. Now, we consider the two cases from (1):

1. $X \neq X'$. Suppose $X' \not\subseteq X$. Then $(X, Y)w_X \in F_S$, but $(X', Y')w_X \notin F_S$, since $X' \setminus X$ is a non-empty subset of $\overline{X}$ and hence gets mapped outside of F. Thus w_X distinguishes these states. If instead we have $X \not\subseteq X'$, then $w_{X'}$ distinguishes the states. Hence if $A_X, A_{X'}$ are atoms, w_X and $w_{X'}$ are in the transition semigroup of $\mathcal{D}$, and the states are distinguishable.

2. $Y \neq Y'$. If $Y' \not\subseteq Y$, then $w_{\overline{Y}}$ distinguishes (X, Y) from (X', Y'); otherwise, $w_{\overline{Y'}}$ distinguishes the states. As before, if $A_{\overline{Y}}, A_{\overline{Y'}}$ are atoms then the states are distinguishable.

□

Theorem 1. *For $n \geqslant 2$, the language L_n of Definition 2 has 2^n atoms and each atom meets the bounds of Proposition 2.*

Proof. The DFA for the atomic intersection A_S is $\mathcal{D}_S = (Q_S, \Sigma, \Delta, (S, \overline{S}), F_S)$, where $F_S = \{(X, Y) \mid X \subseteq \{n\}, Y \subseteq Q_n \setminus \{n\}\}$. By Proposition 1, the transition semigroup of $\mathcal{D}_n$ consists of all n^n transformations of the state set Q_n. Hence $(S, \overline{S})$ can be mapped to a final state in F_S by a transformation that sends S to $\{n\}$ and $\overline{S}$ to $\{1\}$. It follows that all 2^n atomic intersections A_S, $S \subseteq Q_n$ are atoms. By the Distinguishability Lemma, all distinct states in $\mathcal{D}_S$ are distinguishable. It suffices to prove the number of reachable states in each $\mathcal{D}_S$ meets the bounds.

If $S = Q_n$, then the DFA $\mathcal{D}_S$ of A_S has initial state $(Q_n, \emptyset)$. The reachable states of $\mathcal{D}_S$ are of the form $(X, \emptyset)$, where X is the image of Q_n under some transformation in the transition semigroup. Since we have all transformations, we can reach all $2^n - 1$ states $(X, \emptyset)$, $\emptyset \subsetneq X \subseteq Q_n$. For $S = \emptyset$ a similar argument works.

If $\emptyset \subsetneq S \subsetneq Q_n$, then for any state (X, Y) with $1 \leqslant X \leqslant |S|$, $1 \leqslant Y \leqslant n - |S|$ and $X \cap Y = \emptyset$, we can find a transformation mapping S onto X and $\overline{S}$ onto Y. So all these states are reachable, and there are $\sum_{x=1}^{|S|} \sum_{y=1}^{n-|S|} \binom{n}{x}\binom{n-x}{y}$ of them. In addition, $\perp$ is reachable from $(S, \overline{S})$ by the constant transformation $(Q_n \to 1)$ and so the bound is met. □

5 Complexity of Atoms in Right Ideals

If L is a right ideal, one of its quotients is Σ^*; by convention we assume that $K_n = \Sigma^*$. In any atom A_S the quotient K_n must be uncomplemented, that is, we must have $n \in S$. Thus $A_\emptyset$ is not an atom. The results of this section were stated in [4] without proof; for completeness we include the proofs.

Proposition 3. *Suppose L is a right ideal with $n \geqslant 1$ quotients. Then L has at most 2^{n-1} atoms. The complexity $\kappa(A_S)$ of atom A_S satisfies*

$$\kappa(A_S) \leqslant \begin{cases} 2^{n-1}, & \text{if } S = Q_n; \\ 1 + \sum_{x=1}^{|S|} \sum_{y=1}^{n-|S|} \binom{n-1}{x-1}\binom{n-x}{y}, & \text{if } \emptyset \subsetneq S \subsetneq Q_n. \end{cases} \tag{1}$$

Proof. Let A_S be an atom. Since $w^{-1}\Sigma^* = \Sigma^*$ for all $w \in \Sigma^*$, $w^{-1}A_S$ always has K_n uncomplemented; so if (X, Y) corresponds to $w^{-1}A_S$, then $n \in X$. Since the number of subsets S of Q_n containing n is 2^{n-1}, there are at most that many atoms. Consider two cases:

1. $S = Q_n$. Then $w^{-1}L = \bigcap_{i\in X} K_i$, and each such quotient of A_S is represented by $(X, \emptyset)$, where $1 \leqslant |X| \leqslant n$. Since n is always in X, there are at most 2^{n-1} quotients of this atom.

2. $\emptyset \subsetneq S \subsetneq Q_n$. Then $w^{-1}A_S = \bigcap_{i\in X} K_i \cap \bigcap_{i\in Y} \overline{K_i}$, where $1 \leqslant |X| \leqslant |S|$ and $1 \leqslant |Y| \leqslant n - |S|$. We know that if $X \cap Y \neq \emptyset$, then $w^{-1}A_S = \emptyset$. Thus we are looking for pairs (X, Y) such that $n \in X$ and $X \cap Y = \emptyset$. To get X we take n and choose $|X| - 1$ elements from $Q_n \setminus \{n\}$, and then to get Y we take $|Y|$ elements from $Q_n \setminus X$. The number of such pairs is $\sum_{x=1}^{|S|} \sum_{y=1}^{n-|S|} \binom{n-1}{x-1}\binom{n-x}{y}$. Adding the empty quotient we have our bound.

□

For $n = 1$, $L = \Sigma^*$ is a right ideal with one atom of complexity 1. For $n = 2$, $L = aa^*$ is a right ideal with two atoms L and $\overline{L}$ of complexity 2. It was shown in [4] that the languages of the DFAs of Definition 3 are "most complex" amongst right ideals, in the sense that they meet all the complexity bounds for common operations, but no proof of atom complexity was given. We include this proof here.

Definition 3. *For $n \geqslant 3$, let $\mathcal{D}_n = (Q_n, \Sigma, \delta_n, 1, \{n\})$, where $\Sigma = \{a, b, c, d\}$, and δ_n is defined by $a = (1, \ldots, n-1)$, $b = (2, \ldots, n-1)$, $c = (n-1 \to 1)$ and $d = (n-1 \to n)$. Let L_n be the language accepted by $\mathcal{D}_n$. If $n = 3$, b is not needed; hence $\Sigma = \{a, c, d\}$ suffices. Also, let $L_2 = aa^*$ and $L_1 = a^*$.*

Theorem 2. *For $n \geqslant 1$, the language L_n of Definition 3 is a right ideal that has 2^{n-1} atoms and each atom meets the bounds of Proposition 3.*

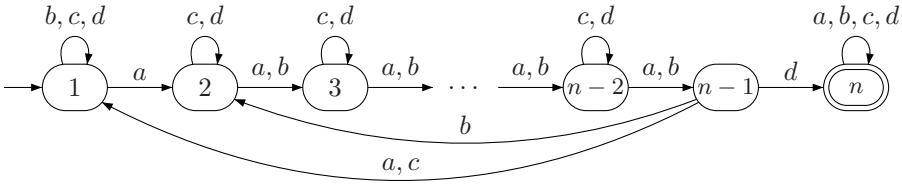

Figure 2: DFA of a right ideal whose atoms meet the bounds.

Proof. The cases $n < 3$ are easily verified; hence assume $n \geqslant 3$. By Proposition 1, the transformations $\{a, b, c\}$ restricted to Q_{n-1} generate all transformations of Q_{n-1}. When d is included, we get all transformations of Q_n that fix n. For $S \subseteq Q_n$, $n \in S$, consider the DFA $\mathcal{D}_S$, which has initial state $(S, \overline{S})$. There is a transformation of Q_n fixing n that sends $(S, \overline{S})$ to the final state $(\{n\}, \{1\})$. Hence A_S is an atom if $n \in S$, and so L_n has 2^{n-1} atoms.

We now count reachable and distinguishable states in the DFA of each atom. Suppose $S = Q_n$. The initial state of $\mathcal{D}_S$ is $(Q_n, \emptyset)$; by transformations that fix n, we can reach any state $(X, \emptyset)$ with $\{n\} \subseteq X \subseteq Q_n$. There are 2^{n-1} such states, and since A_X is an atom if $n \in X$, all of them are distinguishable by the Distinguishability Lemma.

Suppose $\emptyset \subsetneq S \subsetneq Q_n$. From the initial state $(S, \overline{S})$, by transformations that fix n we can reach any (X, Y) with $1 \leqslant |X| \leqslant |S|$, $1 \leqslant |Y| \leqslant n - |S|$, $n \in X$ and $X \cap Y = \emptyset$. There are $\sum_{x=1}^{|S|} \sum_{y=1}^{n-|S|} \binom{n-1}{x-1}\binom{n-x}{y}$ such states. For all such states (X, Y), we have $n \in X$ and $n \in \overline{Y}$, so A_X and $A_{\overline{Y}}$ are both atoms; hence by the Distinguishability Lemma, all of these states are distinguishable from each other and from $\perp$. The state $\perp$ is also reachable by the constant transformation $(Q_n \to n)$, and so the bound is met.

□

6 Complexity of Atoms in Left Ideals

If L is a left ideal, then $L = \Sigma^* L$, and $w^{-1}L$ contains L for every $w \in \Sigma^*$. By convention we let $L = K_1$.

Proposition 4. *Suppose L is a left ideal with $n \geqslant 2$ quotients. Then L has at most $2^{n-1} + 1$ atoms. The complexity $\kappa(A_S)$ of atom A_S satisfies*

$$\kappa(A_S) \begin{cases} = n, & \text{if } S = Q_n; \\ \leqslant 2^{n-1}, & \text{if } S = \emptyset; \\ \leqslant 1 + \sum_{x=1}^{|S|} \sum_{y=1}^{n-|S|} \binom{n-1}{x}\binom{n-x-1}{y-1}, & \text{otherwise.} \end{cases} \tag{2}$$

Proof. Consider the atomic intersections A_S such that $1 \in S$; then $\bigcap_{i \in S} K_i = L$ (since every quotient contains L), and there are two possibilities: Either $S = Q_n$,

in which case $A_S = A_{Q_n} = \bigcap_{i \in Q_n} K_i = L$, or there is at least one quotient, say K_i which is complemented. Since K_i contains L, it can be expressed as $K_i = L \cup M_i$, where $L \cap M_i = \emptyset$. Then the intersection has the term $L \cap \overline{(L \cup M_i)} = \emptyset$, and A_S is not an atom. Thus for A_S to be an atom, either $1 \notin S$ or $S = Q_n$. Hence there are at most $2^{n-1} + 1$ atoms.

For atom complexity, consider the following cases:

1. $S = Q_n$. Then $A_{Q_n} = L$, and the complexity of A_{Q_n} is precisely n.

2. $S = \emptyset$. Now $A_\emptyset = \bigcap_{i \in Q_n} \overline{K_i}$, and every quotient of $A_\emptyset$ is an intersection $\bigcap_{i \in Y} \overline{K_i}$, where $1 \leqslant |Y| \leqslant |Q_n|$. There are $2^n - 1$ such intersections, but consider any quotient $K_i \neq L$ of a left ideal; it can be expressed as $K_i = L \cup M_i$, where $L \cap M_i = \emptyset$. We have

$$\overline{K_1} \cap \overline{K_i} = \overline{L} \cap \overline{L \cup M_i} = \overline{L} \cap \overline{L} \cap \overline{M_i} = \overline{L} \cap \overline{M_i} = \overline{K_i}.$$

   Thus every intersection $\bigcap_{i \in Y} \overline{K_i}$ which has $Y \neq \emptyset$ and does not have $\overline{K_1}$ as a term defines the same language as $\overline{K_1} \cap \bigcap_{i \in Y} \overline{K_i}$. There are $2^{n-1} - 1$ such intersections. Adding 1 for the intersection which just has the single term $\overline{K_1}$, we get our bound 2^{n-1}.

3. $\emptyset \subsetneq S \subsetneq Q_n$. Then $A_S = \bigcap_{i \in S} K_i \cap \bigcap_{i \in \overline{S}} \overline{K_i}$, where neither S nor $\overline{S}$ is empty. If $1 \in S$ then A_S is not an atom, so assume $1 \notin S$. Every quotient of A_S has the form $w^{-1}A_S = \bigcap_{i \in X} K_i \cap \bigcap_{i \in Y} \overline{K_i}$, where $1 \leqslant |X| \leqslant |S|$ and $1 \leqslant |Y| \leqslant n - |S|$.

   a) $1 \in X$. We claim that $w^{-1}A_S = \emptyset$ for all $w \in \Sigma^*$. For suppose that there is a term K_i, $i \in S$, and a word $w \in \Sigma^*$ such that $w^{-1}K_i = K_1$. Since $K_1 \subseteq K_i$, we have $w^{-1}K_1 \subseteq w^{-1}K_i = K_1$. Since also $K_1 \subseteq w^{-1}K_1$ because L is a left ideal, we have $w^{-1}K_1 = K_1$. But $1 \in \overline{S}$, so $w^{-1}\left(\bigcap_{i \in \overline{S}} \overline{K_i}\right) = \bigcap_{i \in Y} \overline{K_i}$ has $w^{-1}\overline{K_1} = \overline{K_1}$ as a term. Thus $1 \in Y$, which means $X \cap Y \neq \emptyset$. Hence $w^{-1}A_S = \emptyset$.

   b) $1 \notin X$. We are looking for pairs (X, Y) such that $X \cap Y = \emptyset$. As we argued in (2), $\overline{K_1} \cap \overline{K_i} = \overline{K_i}$ for each i, so we can assume without loss of generality that $1 \in Y$. To get X we choose $|X|$ elements from $Q_n \setminus \{1\}$ and to get Y we take $\{1\}$ and choose $|Y|-1$ elements from $(Q_n \setminus X) \setminus \{1\}$. The number of such pairs is $\sum_{x=1}^{|S|} \sum_{y=1}^{n-|S|} \binom{n-1}{x} \binom{n-x-1}{y-1}$.

   Adding 1 for the empty quotient we have our bound.

□

Next we compare the bounds for left ideals with those for right ideals. To calculate the number of pairs (X, Y) such that $n \in X$ and $X \cap Y = \emptyset$ for right

ideals, we can first choose Y from $Q_n \setminus \{n\}$, and then choose X by taking n and choosing $|X|-1$ elements from $(Q_n \setminus Y) \setminus \{n\}$. The number of such pairs is

$$1 + \sum_{y=1}^{n-|S|} \sum_{x=1}^{|S|} \binom{n-1}{y} \binom{n-y-1}{x-1}.$$

If we interchange x and y we note that this is precisely the number of pairs (X, Y) such that $1 \in Y$ and $X \cap Y = \emptyset$ for an atom of a left ideal with a basis of size $n - |S|$. Thus we have

Remark 1. Let R be a right ideal of complexity n and let A_S be an atom of R, where $\emptyset \subsetneq S \subsetneq Q_n$. Let L be a left ideal of complexity n and let $A'_{\overline{S}}$ be an atom of L. The upper bounds on the complexities of A_S and $A'_{\overline{S}}$ are equal.

Now we consider the question of tightness of the bounds in Proposition 4. For $n = 1$, $L = \Sigma^*$ is a left ideal with one atom of complexity 1; so the bound of Proposition 4 does not hold.

The DFAs of Definition 4 and Figure 3 were introduced in [10]. It was shown in [7] that the languages of these DFAs have the largest syntactic semigroups amongst left ideals of complexity n. Moreover, it was shown in [5] that these languages also meet the bounds on the quotient complexity of boolean operations, concatenation and star. Together with our result about the number of atoms and their complexity, this shows that these languages are "most complex" left ideals.

Definition 4. *For $n \geqslant 3$, let $\mathcal{D}_n = (Q_n, \Sigma, \delta_n, 1, \{n\})$, where $\Sigma = \{a, b, c, d, e\}$, and δ_n is defined by $a = (2, \ldots, n)$, $b = (2, 3)$, $c = (n \to 2)$, $d = (n \to 1)$, and $e = (Q_n \to 2)$. If $n = 3$, inputs a and b coincide; hence $\Sigma = \{a, c, d, e\}$ suffices. Also, let $\mathcal{D}_2 = (Q_2, \{a, b, c\}, \delta_2, 1, \{2\})$, where $a = \mathbf{1}$, $b = (Q_2 \to 2)$, $c = (Q_2 \to 1)$. Let L_n be the language accepted by $\mathcal{D}_n$; we have $L_2 = \Sigma^* b (a \cup b)^*$.*

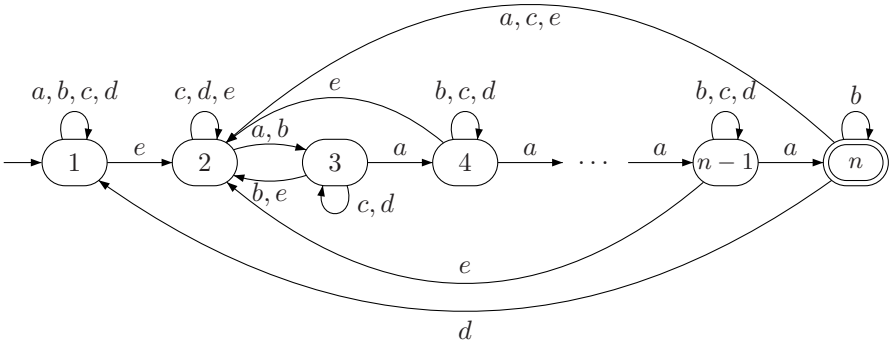

Figure 3: DFA of a left ideal whose atoms meet the bounds.

Theorem 3. *For $n \geqslant 2$, the language L_n of Definition 4 is a left ideal that has $2^{n-1}+1$ atoms and each atom meets the bounds of Proposition 4.*

Proof. It was proved in [10] that L_n is a left ideal of complexity n. The case $n = 2$ is easily verified; hence assume $n \geqslant 3$. It was proved in [7] that the transition semigroup of $\mathcal{D}_n$ contains all transformations of Q_n that fix 1 and all constant transformations. Recall that if A_S is an atom of a left ideal, then either $S = Q_n$ or $1 \notin S$. For all S with $1 \notin S$, from $(S, \overline{S})$ we can reach the final state $(\{n\}, \{1\})$ of $\mathcal{D}_S$ (or $(\emptyset, \{1\})$ for $S = \emptyset$) by transformations that fix 1. For $S = Q_n$, let $w = (Q_n \to n)$; then $(Q_n, \emptyset)w = (\{n\}, \emptyset)$ is final in $\mathcal{D}_S$. Hence if $S = Q_n$ or $1 \notin S$, then A_S is an atom of L_n, and so L has $2^{n-1}+1$ atoms.

We now count reachable and distinguishable states in the DFA of each atom. We know that A_{Q_n} has complexity n for all left ideals, so assume $1 \notin S$. If $S = \emptyset$, the initial state of $\mathcal{D}_S$ is $(\emptyset, Q_n)$. By transformations that fix 1 we can reach $(\emptyset, Y)$ for all Y with $\{1\} \subseteq Y \subseteq Q_n$. There are 2^{n-1} of these states. Since $\overline{Y}$ does not contain 1, $A_{\overline{Y}}$ is an atom, so all of these states are distinguishable by the Distinguishability Lemma.

If $\emptyset \subsetneq S \subsetneq Q_n$, the initial state of $\mathcal{D}_S$ is $(S, \overline{S})$. Since $1 \notin S$, by transformations that fix 1, we can reach any state (X, Y) with $1 \leqslant |X| \leqslant |S|$, $1 \leqslant |Y| \leqslant n - |S|$, $1 \notin X$, $1 \in Y$, and $X \cap Y = \emptyset$. There are $\sum_{x=1}^{|S|} \sum_{y=1}^{n-|S|} \binom{n-1}{x}\binom{n-x-1}{y-1}$ such states. They are all distinguishable from each other and from $\perp$ by the Distinguishability Lemma, since $1 \notin X$, $1 \in Y$ imply that A_X and $A_{\overline{Y}}$ are both atoms. We can also reach $\perp$ from $(S, \overline{S})$ by any constant transformation, and so the bound is met. □

7 Complexity of Atoms in Two-Sided Ideals

7.1 Upper Bounds

A language is a two-sided ideal if it is both a right ideal and a left ideal.

Proposition 5. *Suppose L is a two-sided ideal with $n \geqslant 2$ quotients. Then L has at most $2^{n-2}+1$ atoms. The complexity $\kappa(A_S)$ of atom A_S satisfies*

$$\kappa(A_S) \begin{cases} = n, & \text{if } S = Q_n; \\ \leqslant 2^{n-2} + n - 1, & \text{if } S = Q_n \setminus \{1\}; \\ \leqslant 1 + \sum_{x=1}^{|S|} \sum_{y=1}^{n-|S|} \binom{n-2}{x-1}\binom{n-x-1}{y-1}, & \text{otherwise.} \end{cases} \tag{3}$$

Proof. Since L is a left ideal, A_S is an atom only if $S = Q_n$ or $S \subseteq Q_n \setminus \{1\}$; since L is a right ideal we must also have $n \in S$. This gives our upper bound of $2^{n-2}+1$ atoms.

We know that A_{Q_n} has complexity n since this is true for left ideals. Since L is a right ideal, $A_\emptyset$ is not an atom, so we can assume $S \neq \emptyset$.

Suppose A_S is an atom of L, with $S \neq Q_n$ and $S \neq Q_n \setminus \{1\}$. We proved for left ideals that the number of distinct non-empty quotients of A_S is bounded by the number of pairs (X, Y), $1 \leqslant |X| \leqslant |S|$, $1 \leqslant |Y| \leqslant n - |S|$, $1 \notin X$, $1 \in Y$, $X \cap Y = \emptyset$.

Since L is a right ideal, we must also have $n \in X$ and $n \notin Y$. There are $\binom{n-2}{|X|-1}$ possibilities for X, since X must contain n and the remaining $|X|-1$ elements are taken from $Q_n \setminus \{1,n\}$. If X is fixed, there are $\binom{n-|X|-1}{|Y|-1}$ possibilities for Y, since Y must contain 1 and the remaining $|Y|-1$ elements are taken from $(Q_n \setminus X) \setminus \{1\}$. Since $Q_n \setminus X$ always contains 1, the size of $(Q_n \setminus X) \setminus \{1\}$ is always $n-|X|-1$. Summing over the possible sizes of X and Y and adding 1 for the empty quotient, we get the required bound.

This leaves the case of $S = Q_n \setminus \{1\}$. Each quotient of A_S has the form

$$w^{-1}A_S = \left(\bigcap_{i \in X} K_i\right) \cap \overline{K_j},$$

where $K_j = w^{-1}K_1 = w^{-1}L$, and $n \in X$. We can view the non-empty quotients as states $(X, \{j\})$ of the DFA $\mathcal{D}_S$ for A_S, where $\mathcal{D}$ is a minimal DFA for L. We must have $n \in X$ and $X \cap \{j\} = \emptyset$, and so $j \notin X$.

For each p in Q_n, define the set $S(p) = \{q \in Q_n \mid K_p \subsetneq K_q\}$. The elements of $S(p)$ are called the *successors* of p. Note that p is not a successor of itself. We claim that if the quotient $w^{-1}A_S$ is non-empty and the corresponding state of $\mathcal{D}_S$ is $(X, \{j\})$, then $X \subseteq S(j)$.

To see this, note that since L is a left ideal, we have $L \subseteq K_i$ for all $i \in Q_n$. It follows that $w^{-1}L = K_j \subseteq w^{-1}K_i$ for all $i \in Q_n$. Thus in the formula for $w^{-1}A_S$ above, we have $K_j \subseteq K_i$ for all $i \in X$. But if $K_j = K_i$ for any $i \in X$, then $w^{-1}A_S$ is empty. Thus $K_j \subsetneq K_i$ for all $i \in X$, which implies that $X \subseteq S(j)$.

X must contain n, since L is a right ideal. Thus for each j, there are at most $2^{|S(j)|-1}$ states $(X, \{j\})$. The index j can range from 1 to $n-1$; we cannot have $j = n$ since $n \in X$ but $j \notin X$. This gives an upper bound of $\sum_{j=1}^{n-1} 2^{|S(j)|-1}$ for the number of non-empty quotients.

This bound is not tight, so we refine it by considering the distinguishability relations between states of $\mathcal{D}_S$. Choose $i \neq n \in S(j)$ and a non-empty set $Y \subseteq S(i) \setminus \{n\}$. Then $K_i \subsetneq K_q$ for all $q \in Y$, so we have $K_i \cap \left(\bigcap_{q \in Y} K_i\right) = K_i$. This means $(\{i,n\}, \{j\})$ is indistinguishable from $(Y \cup \{i,n\}, \{j\})$. Since Y is non-empty and does not contain n, there are at most $2^{|S(i)|-1} - 1$ possibilities for Y.

From this we get a new upper bound for the number of distinguishable states $(X, \{j\})$ for a fixed j, as follows: first take our previous bound of $2^{|S(j)|-1}$. Then for each $i \neq n \in S(j)$, subtract $2^{|S(i)|-1} - 1$ to account for the states $(Y \cup \{i,n\}, \{j\})$ that are equivalent to $(\{i,n\}, \{j\})$. Our new bound is

$$2^{|S(j)|-1} - \sum_{\substack{i \in S(j) \\ i \neq n}} (2^{|S(i)|-1} - 1) = (|S(j)| - 1) + 2^{|S(j)|-1} - \sum_{\substack{i \in S(j) \\ i \neq n}} 2^{|S(i)|-1}.$$

Summing over all possible values of j, and adding 1 for the empty quotient, we get

the following bound on the complexity of A_S:

$$1+\sum_{j=1}^{n-1}\left((|S(j)|-1)+2^{|S(j)|-1}-\sum_{\substack{i\in S(j)\\ i\neq n}}2^{|S(i)|-1}\right).$$

Noting that $S(1)=\{2,\ldots,n\}$ and $|S(1)|=n-1$, we pull out the $j=1$ case from the outermost summation:

$$1+(n-2)+2^{n-2}-\sum_{\substack{i\in S(1)\\ i\neq n}}2^{|S(i)|-1}+\sum_{j=2}^{n-1}\left((|S(j)|-1)+2^{|S(j)|-1}-\sum_{\substack{i\in S(j)\\ i\neq n}}2^{|S(i)|-1}\right).$$

Observe that $1+(n-2)+2^{n-2}$ is equal to $2^{n-2}+n-1$, the bound we are trying to prove. We will show that the value of the rest of this formula is always less than or equal to zero. We pull $\sum_{j=2}^{n-1}2^{|S(j)|-1}$ out to the front:

$$2^{n-2}+n-1+\sum_{j=2}^{n-1}2^{|S(j)|-1}-\sum_{\substack{i\in S(1)\\ i\neq n}}2^{|S(i)|-1}+\sum_{j=2}^{n-1}\left((|S(j)|-1)-\sum_{\substack{i\in S(j)\\ i\neq n}}2^{|S(i)|-1}\right).$$

Note that $\sum_{j=2}^{n-1}2^{|S(j)|-1}=\sum_{\substack{i\in S(1)\\ i\neq n}}2^{|S(i)|-1}$, so cancellation occurs:

$$2^{n-2}+n-1+\sum_{j=2}^{n-1}\left((|S(j)|-1)-\sum_{\substack{i\in S(j)\\ i\neq n}}2^{|S(i)|-1}\right).$$

Now, the value of the innermost summation is always greater than or equal to $|S(j)|-1$: for each $i\in S(j)$, $i\neq n$, we know that n is a successor of i, and hence $S(i)\geqslant 1$ and $2^{|S(i)|-1}\geqslant 1$. Thus the value of the outermost summation is always less than or equal to zero. It follows that the number of quotients of A_S is at most $2^{n-2}+n-1$. □

Next we address the question of tightness of the bounds for two-sided ideals. For $n=1$, $L=\Sigma^*$ is a two-sided ideal with one atom of complexity 1; so the bound of Proposition 5 does not hold.

The DFAs of Definition 5 and Figure 4 were introduced in [10]. It was shown in [7] that these languages have the largest syntactic semigroups amongst two-sided ideals of complexity n. Moreover, it was shown in [5] that they also meet the bounds on the quotient complexity of boolean operations, concatenation and star. Together with our result about the number of atoms and their complexity, this shows that these languages are "most complex" two-sided ideals.

Definition 5. *Let $n \geqslant 4$, and let $\mathcal{D}_n = (Q_n, \Sigma, \delta_n, 1, \{n\})$ be the DFA with $\Sigma = \{a, b, c, d, e, f\}$, $a = (2, 3, \ldots, n-1)$, $b = (2, 3)$, $c = (n-1 \to 2)$, $d = (n-1 \to 1)$, $e = (Q_{n-1} \to 2)$, and $f = (2 \to n)$. For $n = 4$, inputs a and b coincide. Also, let $\mathcal{D}_3 = (Q_3, \{a, b, c\}, \delta_3, 1, \{3\})$, where $a = \mathbf{1}$, $b = (Q_2 \to 2)$, $c = (2 \to 3)$, and let $\mathcal{D}_2 = (Q_2, \{a, b, c\}, \delta_2, 1, \{2\})$, where $a = \mathbf{1}$, $b = (Q_2 \to 2)$, $c = (Q_2 \to 1)$. Let L_n be the language accepted by $\mathcal{D}_n$.*

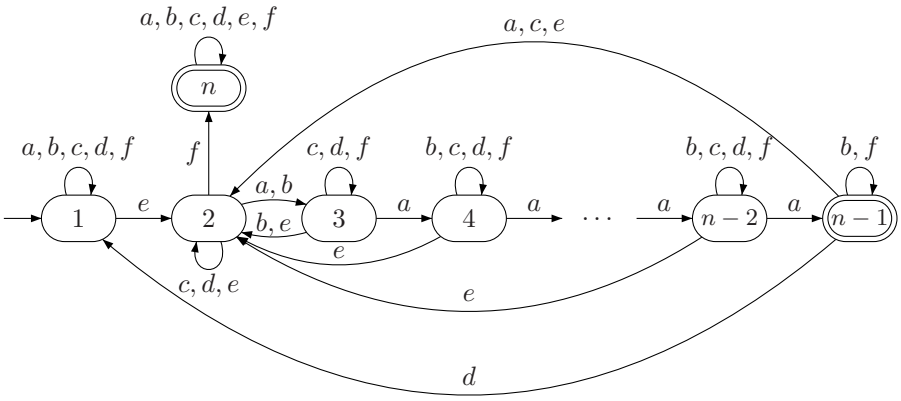

Figure 4: DFA of a two-sided ideal whose atoms meet the bounds.

Theorem 4. *For $n \geqslant 2$, the language L_n of Definition 5 is a two-sided ideal that has $2^{n-2} + 1$ atoms and each atom meets the bounds of Proposition 5.*

Proof. It was proved in [10] that L_n is a two-sided ideal of complexity n. The cases with $n < 4$ are easily verified; hence assume $n \geqslant 4$.

The following observations were made in [7]: Transformations $\{a, b, c\}$ restricted to $Q_n \setminus \{1, n\}$ generate all the transformations of $\{2, \ldots, n-1\}$. Together with d and f, they generate all transformations of Q_n that fix 1 and n. Also, we have $ef = (Q_n \to n)$.

Recall that if A_S is an atom of a two-sided ideal, then $n \in S$, and either $S = Q_n$ or $1 \notin S$. We know A_{Q_n} is an atom of complexity n for all left ideals (and hence all two-sided ideals), so assume $n \in S$, $1 \notin S$. Then $1 \in \overline{S}$, and so from state $(S, \overline{S})$ in $\mathcal{D}_S$ we can reach the final state $(\{n\}, \{1\})$ by transformations that fix 1 and n. Hence A_S is an atom for every S with $n \in S$, $1 \notin S$. There are 2^{n-2} of these atoms, as well as the atom A_{Q_n}, for a total of $2^{n-2} + 1$.

Consider the atom A_S for $S \neq Q_n$ and $S \neq Q_n \setminus \{1\}$. In the DFA $\mathcal{D}_S$, the initial state is $(S, \overline{S})$, and we have $n \in S$, $1 \notin S$. By transformations that fix 1 and n, we can reach (X, Y) for all $X, Y \subseteq Q_n$ such that $n \in X$, $1 \in Y$, $X \cap Y = \emptyset$, $1 \leqslant |X| \leqslant |S|$, $1 \leqslant |Y| \leqslant n - |S|$. There are $\sum_{x=1}^{|S|} \sum_{y=1}^{n-|S|} \binom{n-2}{x-1}\binom{n-x-1}{y-1}$ such states. Since $n \in X$, $1 \notin X$ and $n \in \overline{Y}$, $1 \notin \overline{Y}$ we see that A_X and $A_{\overline{Y}}$ are atoms. Hence by the Distinguishability Lemma, all of these states are distinguishable from

each other and from $\perp$. Since $S \neq \emptyset$, we can reach $\perp$ from $(S, \overline{S})$ by $ef = (Q_n \to n)$. Hence the bound is met.

It remains to show that the complexity of A_S, $S = Q_n \setminus \{1\}$ also meets the bound. The initial state of $\mathcal{D}_S$ is $(\{2, \ldots, n\}, \{1\})$. By transformations that fix 1 and n, we can reach all 2^{n-2} states of the form $(X, \{1\})$ with $\{n\} \subseteq X \subseteq Q_n \setminus \{1\}$. From $(\{n\}, \{1\})$, we can reach $n-2$ additional states $(\{n\}, \{i\})$ for $2 \leqslant i \leqslant n-1$ by ea^{i-2}. Finally, we can reach the sink state $\perp$ from the initial state by $ef = (Q_n \to n)$. This gives a total of $2^{n-2}+n-1$ reachable states, which matches the upper bound.

To see these states are distinguishable, note that A_X is an atom if $\{n\} \subseteq X \subseteq Q_n \setminus \{1\}$. Also, $A_{\overline{\{1\}}} = A_{Q_n \setminus \{1\}}$ is an atom. Hence by the Distinguishability Lemma, all states of the form $(X, \{1\})$ are distinguishable from each other and from $\perp$. Also, $(\{n\}, \{i\})$ is distinguished from $(\{n\}, \{j\})$ by $a^{n-i}f$, which sends the former state to the non-final state $\perp$, but sends the latter to some final state $(\{n\}, \{k\})$ with $k \neq 2$. And each $(\{n\}, \{j\})$, $1 \leqslant j \leqslant n-1$ is a final state, so it is distinguishable from all states of the form $(X, \{1\})$, $X \neq \{n\}$ and from $\perp$, since they are not final. Hence all $2^{n-2} + n - 1$ reachable states are distinguishable. □

8 Some Numerical Results

The following tables compare the maximal complexities for atoms A_S of two-sided ideals (first entry), left ideals (second entry) and regular languages (third entry) with complexity n. Right ideals are omitted because their complexities are essentially the same as those of left ideals, by Remark 1. When the maximal complexity is undefined (e.g., because no languages in a class have atoms A_S for a particular size of S) this is indicated by an asterisk. The maximum values for each n are in boldface. The n^{th} entry in the *ratio* row shows the approximate value of m_n/m_{n-1}, where m_i is the i^{th} entry in the *max* row. It has been shown by Diekert and Walter [11] that the ratio converges exponentially fast to 3 for the class of regular languages and for all three classes of ideal languages.

n	1	2	3	4	5	$\cdots$
$\lvert S\rvert = 0$	$*$/**1**/**1**	$*$/**2**/**3**	$*$/4/7	$*$/8/15	$*$/16/31	$\cdots$
$\lvert S\rvert = 1$	**1**/**1**/**1**	**2**/**2**/**3**	3/**5**/**10**	5/13/29	9/33/76	$\cdots$
$\lvert S\rvert = 2$		**2**/**2**/**3**	**4**/4/**10**	**8**/**16**/**43**	**20**/**53**/**141**	$\cdots$
$\lvert S\rvert = 3$			3/3/7	7/8/29	**20**/43/**141**	$\cdots$
$\lvert S\rvert = 4$				4/4/15	12/16/76	$\cdots$
$\lvert S\rvert = 5$					5/5/31	$\cdots$
max	1/1/1	2/2/3	4/5/10	8/16/43	20/53/141	$\cdots$
ratio	–	2.00/2.00/3.00	2.00/2.50/3.33	2.00/3.20/4.30	2.50/3.31/3.28	$\cdots$

n	6	7	8	9
$\|S\| = 0$	$*/32/63$	$*/64/127$	$*/128/255$	$*/256/511$
$\|S\| = 1$	$17/81/187$	$33/193/442$	$65/449/1,017$	$129/1,025/2,296$
$\|S\| = 2$	$48/156/406$	$112/427/1,086$	$256/1,114/2,773$	$576/2,809/6,859$
$\|S\| = 3$	$\mathbf{64/166/501}$	$\mathbf{182/542/1,548}$	$484/1,611/4,425$	$1,234/4,517/12,043$
$\|S\| = 4$	$48/106/406$	$\mathbf{182}/462/\mathbf{1,548}$	$\mathbf{584/1,646/5,083}$	$\mathbf{1,710/5,245/15,361}$
$\|S\| = 5$	$21/32/187$	$112/249/1,086$	$484/1,205/4,425$	$\mathbf{1,710}/4,643/\mathbf{15,361}$
$\|S\| = 6$	$6/6/63$	$38/64/442$	$256/568/2,773$	$1,234/3,019/12,043$
$\|S\| = 7$		$7/7/127$	$71/128/1,017$	$576/1,271/6,859$
$\|S\| = 8$			$8/8/255$	$136/256/2,296$
$\|S\| = 9$				$9/9/511$
max	$64/166/501$	$182/542/1,548$	$584/1,646/5,083$	$1,710/5,245/15,361$
ratio	$3.20/3.13/3.55$	$2.84/3.27/3.09$	$3.21/3.04/3.28$	$2.93/3.19/3.02$

9 Conclusions

We have derived tight upper bounds for the number of atoms and quotient complexity of atoms in right, left and two-sided regular ideal languages. The recently discovered relationship between atoms and the Myhill and Nerode congruence classes opens up many interesting research questions. The quotient complexity of a language is equal to the number of Nerode classes, and the number of Myhill classes has also been used as a measure of complexity, called *syntactic complexity* since it is equal to the size of the syntactic semigroup. We can view the number of atoms as a third fundamental measure of complexity for regular languages.

It is known [8] that the number of atoms of a regular language L is equal to the quotient complexity of the *reversal* of L. The quotient complexity of reversal has been studied for various classes of languages in the context of determining the quotient complexity of operations on regular languages. Hence, the maximal number of atoms is known for many language classes.

However, as far as we know the *quotient complexity* of atoms has not been studied outside of regular languages and ideals. For simplicity, let us call the atom congruence the *left congruence*, the Nerode congruence the *right congruence*, and the Myhill congruence the *central congruence*. When computing the quotient complexity of atoms, we are computing the number of *right congruence classes* of each *left congruence class*. We can consider variations of this idea: how many right classes and left classes do the central classes have? How many central classes do the left classes have? These questions are outside the scope of this paper, but we believe they should be investigated.

References

[1] Brzozowski, J. Quotient complexity of regular languages. *J. Autom. Lang. Comb.*, 15(1/2):71–89, 2010.

[2] Brzozowski, J. In search of the most complex regular languages. *Int. J. Found. Comput. Sci.,*, 24(6):691–708, 2013.

[3] Brzozowski, J. and Davies, G. Maximally atomic languages. In Ésik, Z. and Fülöp, Z., editors, *Proceedings of the 14th International Conference on Automata and Formal Languages* (*AFL*), volume 151 of *Electronic Proceedings in Theoretical Computer Science*, pages 151–161, 2014.

[4] Brzozowski, J. and Davies, G. Most complex regular right ideals. In Jürgensen, H. and et al., editors, *Proceedings of the 16th International Workshop on Descriptional Complexity of Formal Systems* (*DCFS*), volume 8614 of *Lecture Notes in Computer Science*, pages 90–101, Berlin/Heidelberg, 2014. Springer.

[5] Brzozowski, J., Davies, S., and Liu, B. Y. V. Most complex regular ideal languages. `http://arxiv.org/abs/1511.00157`, November, 2015.

[6] Brzozowski, J., Jirásková, G., and Li, B. Quotient complexity of ideal languages. *Theoret. Comput. Sci.*, 470:36–52, 2013.

[7] Brzozowski, J. and Szykuła, M. Upper bounds on syntactic complexity of left and two-sided ideals. In Shur, A. M. and Volkov, M. V., editors, *Proceedings of the 18th International Conference on Developments in Language Theory* (*DLT*), volume 8633 of *Lecture Notes in Computer Science*, pages 13–24, Berlin/Heidelberg, 2014. Springer.

[8] Brzozowski, J. and Tamm, H. Quotient complexities of atoms of regular languages. *Int. J. Found. Comput. Sci.*, 24(7):1009–1027, 2013.

[9] Brzozowski, J. and Tamm, H. Theory of átomata. *Theoret. Comput. Sci.*, 539:13–27, 2014.

[10] Brzozowski, J. and Ye, Y. Syntactic complexity of ideal and closed languages. In Mauri, G. and Leporati, A., editors, *Proceedings of the 15th International Conference on Developments in Language Theory* (*DLT*), volume 6795 of *Lecture Notes in Computer Science*, pages 117–128, Berlin/Heidelberg, 2011. Springer.

[11] Diekert, V. and Walter, T. Asymptotic approximation for the quotient complexities of atoms. *Acta Cybernetica*, 22(2):349–357, 2015.

[12] Iván, S. Complexity of atoms, combinatorially. `http://arxiv.org/abs/1404.6632`, 2015.

[13] Myhill, J. Finite automata and representation of events. *Wright Air Development Center Technical Report*, 57–624, 1957.

[14] Nerode, A. Linear automaton transformations. *Proc. Amer. Math. Soc.*, 9:541–544, 1958.

[15] Pin, Jean-Eric. Syntactic semigroups. In *Handbook of Formal Languages, vol. 1: Word, Language, Grammar*, pages 679–746. Springer, New York, NY, USA, 1997.

[16] Yu, Sheng. State complexity of regular languages. *J. Autom. Lang. Comb.*, 6:221–234, 2001.

Received 15th June 2015

Acta Cybernetica **22** (2015) 313–334.

Conditional Lindenmayer Systems with Subregular Conditions: The Extended Case

Jürgen Dassow* and Stefan Rudolf†

In memorian Ferenc Gécseg, a pioneer in Theoretical Computer Science

Abstract

We study the generative power of extended conditional Lindenmayer systems where the conditions are finite, monoidal, combinational, definite, nilpotent, strictly locally (k)-testable, commutative, circular, suffix-closed, star-free, and union-free regular languages. The results correspond to those obtained for conditional context-free languages.

Keywords: Lindenmayer systems, controlled derivations, subregular conditions

1 Introduction

In the theory of formal languages one imposes very often conditions to perform a step in the generation of words. By practical reasons – but also by theoretical considerations – it is very useful that one can check the condition by an efficient procedure. Thus one relates the condition to regular languages, for which the membership problem can be decided in linear time. We mention here as examples regularly controlled context-free grammars, conditional context-free grammars, tree controlled context-free grammars, networks of evolutionary processors with regular filters, and contextual grammars with selection languages (for details see [4], [16], [13], and [14]).

In these cases the process of checking the condition given by a regular language is now very simple and efficient, however, the increase of generative power is considerable (for instance, for the first four devices mentioned above, one has an increase from context-free languages to recursively enumerable languages). Since on the one hand practical requirements do not ask for arbitrary regular languages and on the other hand theoretical studies – for instance proofs – show that only special regular languages are used, it is very natural to study the devices with subregular languages

*Otto-von-Guericke-Universität Magdeburg, Fakultät für Informatik, PSF 4120, D-39016 Magdeburg, Germany, E-mail: dassow@iws.cs.uni-magdeburg.de

†Fliederweg 7b, D–65527 Niedernhausen, Germany

for the control. The effect of using subregular languages defined by combinatorial and algebraic properties to the generative power was investigated in the last two decades (see e. g. [1], [3], and [6]).

In 1968, A. Lindenmayer introduced a new type of formal grammars and languages in order to describe the development of organisms. We refer to [15] as a monograph on Lindenmayer systems and languages.

Also in this area it is necessary to restrict the applicability of tables by biological reasons (e.g. in order to model the change of the seasons, the different development if water is present or not etc.). In a conditional Lindenmayer system a table P can only be applied to a sentential form w if w belongs to a language associated to P. A first variant of such systems was studied in [17].

In [5] the systematic study of conditional Lindenmayer systems where the languages associated to the tables belong to some family of subregular languages was started. In [5], the case of non-extended Lindenmayer systems was investigated. In this paper we continue by the consideration of extended conditional Lindenmayer systems with subregular conditions. We prove that propagating extended conditional Lindenmayer systems with suffix-closed, union-free, star-free, circular, or strictly locally k-testable (for $k \geq 2$) conditions allow further characterizations of the family of context-sensitive languages, whereas the use of monoidal, combinational, definite, nilpotent and strictly-locally 1-testable languages as conditions does not lead to an increase of the power, i. e., one obtains the family of ET0L languages; systems with commutative conditions are as powerful as non-erasing matrix grammars (with appearance checking). For arbitrary Lindenmayer systems (with erasing rules) one gets characterizations of the family of recursively enumerable languages, if the conditions are suffix-closed, union-free, star-free, circular, strictly locally k-testable (for $k \geq 1$), or commutative; for the other families of subregular languages the place in the hierarchy is not determined completely.

2 Definitions

We assume that the reader is familiar with the basic concepts of the theory of formal languages and automata. In this section we only recall some notations and some definitions such that a reader can understand the results. We refer to [16], [15], and [4].

The inclusion of the set X in the set Y is denoted by $X \subseteq Y$. If the inclusion is strict, we write $X \subset Y$.

For an alphabet V, i. e, V is a finite non-empty set, the set of all words and all non-empty words over V are denoted by V^* and V^+, respectively. The empty word is denoted by λ. For a language L, let $\mathrm{alph}(L)$ be the minimal set V such that $L \subseteq V^*$. For a word $w \in V^*$ and a subset C of V, the number of occurrences of letters of C in w is denoted by $\#_C(w)$. If C only consists of a letter a, we write $\#_a(w)$ instead of $\#_{\{a\}}(w)$.

Let $V = \{a_1, a_2, \ldots, a_n\}$ (with a fixed order of the letters of V). Then, for a

word $w \in V^*$, we define the Parikh vector $\pi_V(w)$ of w by

$$\pi_V(w) = (\#_{a_1}(w), \#_{a_2}(w), \ldots, \#_{a_n}(w)).$$

For a language L over V, we set

$$\pi_V(L) = \{\pi_V(w) \mid w \in L\}.$$

A language L over V is called semi-linear if $\pi_V(L)$ is a finite union of sets of the form

$$\{(a_1, a_2, \ldots, a_n) + \sum_{j=1}^{p} \alpha_j(b_{1,j}, b_{2,j}, \ldots, b_{n,j}) \mid \alpha_j \in \mathbb{N}\}.$$

If we consider a primed version $V' = \{a' \mid a \in V\}$ of some alphabet V, then, for a word $w = a_1 a_2 \ldots a_m$ with $a_i \in V$ for $1 \leq i \leq m$, we set $w' = a'_1 a'_2 \ldots a'_m$. Moreover, if U is a subset of V, then we set $U' = \{a' \mid a \in U\}$. Analogous notation we also use for double primed versions of V, etc.

In this paper two languages L_1 and L_2 are considered as equal if they differ at most in the empty word, i. e., $L_1 \setminus \{\lambda\} = L_2 \setminus \{\lambda\}$.

The families of finite, regular, context-free, context-sensitive, and recursively enumerable languages are denoted by *FIN*, *REG*, *CF*, *CS*, and *RE*, respectively.

2.1 Matrix Grammars and Languages

Matrix grammars are an important representant of grammars with controlled derivations. They are equivalent to many other such devices. We recall their definition since we shall show that also some extended conditional Lindenmayer systems are equivalent to matrix grammars.

A *matrix grammar* is a quintuple $G = (N, T, M, S, Q)$ where
- N and T are disjunct alphabets of nonterminals and terminals,
- $M = \{m_1, m_2, \ldots, m_r\}$ is a finite set of finite sequences m_i of context-free rules, i. e.,
$$m_i = (A_{i,1} \to v_{i,1}, A_{i,2} \to v_{i,2}, \ldots A_{i,r_i} \to v_{i,r_i})$$
for $1 \leq i \leq r$ (the elements of M are called matrices),
- S is an element of N, and
- Q is a subset of the productions occurring in the matrices of M

The application of a matrix m_i is defined as a sequential application of the rules of m_i in the given order where a rule of Q can be ignored if its left-hand side does not occur in the current sentential form, i.e., $x \Longrightarrow_{m_i} y$ holds iff there are words w_j, $1 \leq j \leq r_i + 1$, such that $x = w_1$, $y = w_{r_i+1}$ and, for $1 \leq j \leq r_i$,

$$w_j = x_j A_{i,j} y_j \text{ and } w_{j+1} = x_j v_{i,j} y_j$$

or

$$w_j = w_{j+1} \text{ and } A_{i,j} \text{ does not occur in } w_j \text{ and } A_{i,j} \to v_{i.j} \in Q.$$

The language $L(G)$ generated by G consists of all words $z \in T^+$ such that there is a derivation

$$S \Longrightarrow_{m_{i_1}} v_1 \Longrightarrow_{m_{i_2}} v_2 \Longrightarrow_{m_{i_3}} \ldots \Longrightarrow_{m_{i_t}} v_t = z$$

for some $t \geq 1$.

By MAT^λ and MAT we denote the families of languages generated by matrix grammars and matrix grammars without erasing rules, respectively.

It is well-known that

$$CF \subset MAT \subset CS \subset RE = MAT^\lambda.$$

2.2 Subregular Families of Languages

The aim of this section is the definition of the subregular families of languages considered in this paper and the relation between them.

For a language L over V, we set

$$\begin{aligned} \mathrm{Comm}(L) &= \{a_{i_1} \ldots a_{i_n} \mid a_1 \ldots a_n \in L,\ n \geq 1,\ \{i_1, i_2, \ldots, i_n\} = \{1, 2, \ldots, n\}\}, \\ \mathrm{Circ}(L) &= \{vu \mid uv \in L,\ u, v \in V^*\}, \\ \mathrm{Suf}(L) &= \{v \mid uv \in L,\ u, v \in V^*\} \end{aligned}$$

We consider the following restrictions for regular languages. For a language L with $V = \mathrm{alph}(L)$, we say that L is

- *combinational* iff it can be represented in the form $L = V^*A$ for some subset $A \subseteq V$,
- *definite* iff it can be represented in the form $L = A \cup V^*B$ where A and B are finite subsets of V^*,
- *nilpotent* iff L is finite or $V^* \setminus L$ is finite,
- *commutative* iff $L = \mathrm{Comm}(L)$,
- *circular* iff $L = \mathrm{Circ}(L)$,
- *suffix-closed* (or *fully initial* or *multiple-entry* language) iff $\mathrm{Suf}(L) = L$,
- *union-free* iff L can be described by a regular expression which is only built by product and star,
- *star-free* (or *non-counting*) iff L can be described by a regular expression which is built by union, product, and complementation,
- *monoidal* iff $L = V^*$,

For more details on languages of the types defined above we refer to [19], [11], and [18].

In [2], it was shown that a regular language $R \subset V^*$ is commutative if and only if there is a semi-linear set M and $R = \pi_V^{-1}(M)$.

It is obvious that combinational, definite, nilpotent, union-free and star-free languages are regular, whereas non-regular languages of the other types mentioned above exist.

For a natural number $k \geq 1$, a language L is *strictly locally k-testable* iff there are three subsets A, B and C of V^k such that $a_1a_2 \ldots a_n$ with $n \geq k$ and $a_i \in V$,

$1 \leq i \leq n$, belongs to L iff $a_1 a_2 \dots a_k \in A$, $a_{j+1} a_{j+2} \dots a_{j+k} \in B$ for $1 \leq j \leq n-k-1$, and $a_{n-k+1} a_{n-k+2} \dots a_n \in C$. Moreover, a language L is called *strictly locally testable* iff it is strictly locally k-testable for some $k \geq 1$.

Obviously, strictly locally testable languages can be accepted by finite automata, and hence they are regular.

A set $R \subset V^*$ is strictly locally 1-testable if and only if there are sets $A \subseteq V$, $B \subseteq V$, and $C \subseteq V$ such that $R = AC^*B \cup (A \cap B)$ (see for instance [2]).

By $COMB$, DEF, NIL, $COMM$, $CIRC$, SUF, UF, SF, MON, LOC_k, $k \geq 1$, and LOC, we denote the families of all combinational, definite, nilpotent, regular commutative, regular circular, regular suffix-closed, union-free, star-free, monoidal, strictly locally k-testable, and strictly locally testable languages, respectively. We set

$$\begin{aligned} \mathcal{G} \;=\; & \{FIN, MON, COMB, DEF, NIL, COMM, CIRC, SUF, UF, SF, LOC\} \\ & \cup \{LOC_k \mid k \geq 1\}. \end{aligned}$$

The relations between families of $\mathcal{G}$ are investigated e. g. in [12] and [20] and their set-theoretic relations are given in Figure 1.

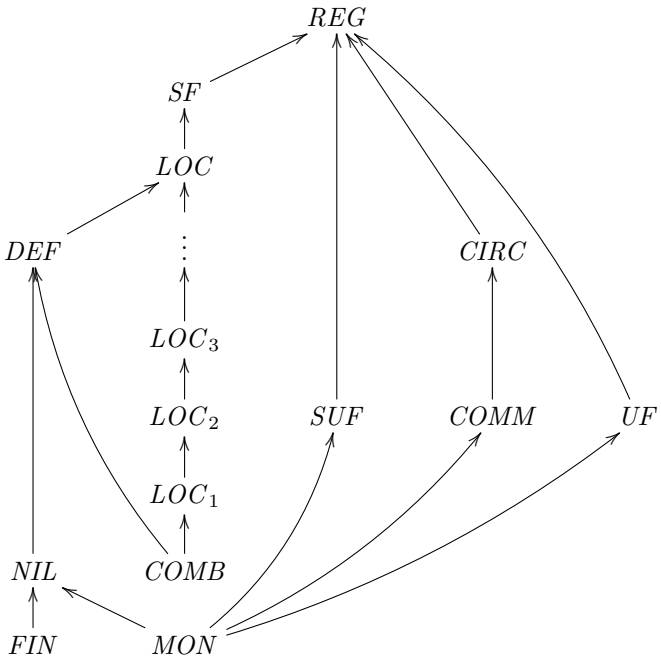

Figure 1: Hierarchy of subregular languages (an arrow from X to Y denotes $X \subset Y$, and if two families are not connected by a directed path then they are incomparable)

Representations of definite automata and definite and nilpotent tree automata and languages were studied by Ferenc Gécseg and coauthors in [7], [8], [9], and [10].

2.3 Extended Conditional Lindenmayer Systems

We start with some definitions concerning Lindenmayer systems and introduce then conditional Lindenmayer systems.

An *extended tabled Lindenmayer system without interaction* (*ET0L system*, for short) is an $(r+3)$-tuple $H = (V, T, P_1, P_2, \ldots, P_r, w)$, where

- V is an alphabet, T is a subset of V,
- for $1 \leq i \leq r$, P_i is a finite set of rules $a \to v$ with $a \in V$ and $v \in V^*$ such that, for any $b \in V$, there is a word v_b with $b \to v_b \in P_i$,
- $w \in V^+$.

The sets P_i, $1 \leq i \leq r$, are called tables. For simplicity, for a table, we shall give only the rules for the letters a for which a rule $a \to w$ with $w \neq a$ exists in the table, i. e., for all letters b, for which no rules are mentioned, there is only the rule $b \to b$ in the table.

For $x \in V^+$ and $y \in V^*$, we say that x derives y in H, written as $x \Longrightarrow_H y$, iff

- $x = a_1 a_2 \ldots a_n$ with $a_i \in V$ for $1 \leq i \leq n$,
- $y = y_1 y_2 \ldots y_n$,
- $a_i \to y_i \in P_j$ for $1 \leq i \leq n$ and some j, $1 \leq j \leq r$.

The language $L(H)$ generated by H is defined as

$$L(H) = \{z \mid z \in T^*,\ w \Longrightarrow_H^* z\}$$

where $\Longrightarrow_H^*$ is the reflexive and transitive closure of $\Longrightarrow_H$.

An ET0L system is called *propagating* if no table contains a rule $a \to \lambda$.

By *ET0L* and *EPT0L*, we denote the families of all languages generated by ET0L systems and propagating ET0L systems, respectively.

It is well-known that the following relation holds

$$CF \subset EPT0L = ET0L \subset MAT.$$

Definition 1. *A conditional ET0L system is an $(n+3)$-tuple*

$$H = (V, T, (P_1, R_1), (P_2, R_2), \ldots, (P_n, R_n), w),$$

where

- *$H' = (V, T, P_1, P_2, \ldots, P_n, w)$ is an ET0L system, and,*
- *for $1 \leq i \leq n$, R_i is a regular language over some alphabet $U \subseteq V$.*

For $x \in V^+$ and $y \in V^*$, we say that x derives y in H, written as $x \Longrightarrow_H y$, if and only if there is a number j, $1 \leq j \leq n$

- *$x = a_1 a_2 \ldots a_t$ with $a_i \in V$ for $1 \leq i \leq t$,*
- *$y = y_1 y_2 \ldots y_t$,*
- *$a_i \to y_i \in P_j$ for $1 \leq i \leq t$, and*
- *$x \in R_j$.*

The language $L(H)$ generated by H is defined as

$$L(H) = \{z \mid z \in T^*,\ w \Longrightarrow_H^* z\}$$

where $\Longrightarrow_H^*$ is the reflexive and transitive closure of $\Longrightarrow_H$.

By definition, in a condition ET0L system, a table P_j is only applicable to a sentential form x, if x belongs to the conditional language R_j associated with P_j.

Example 1. We consider the ET0L system

$$H = (V, \{a, b\}, (P_1, R_1), (P_2, R_2), (P_3, R_3), (P_4, R_4), (P_5, R_5), SD)$$

with

$$\begin{aligned} V &= \{S, A, B_1, B_2, C, D, a, b\}, \\ (P_1, R_1) &= (\{S \to ASC\}, V^*\{D\}), \\ (P_2, R_2) &= (\{S \to AC, D \to \lambda\}, V^*\{D\}), \\ (P_3, R_3) &= (\{A \to Ab, C \to B_1, C \to B_2\}, V^*\{C\}), \\ (P_4, R_4) &= (\{B_1 \to \lambda, B_2 \to C\}, V^*\{B_1\}), \\ (P_5, R_5) &= (\{A \to a\}, V^*\{b\}). \end{aligned}$$

We start with SD, have to apply sometimes P_1 and then once P_2 (the only rules where the words in the associated language end with D). This yields A^nC^n. Now we have to apply P_3 and get $(Ab)^nz$ where z is a word of length n over $\{B_1, B_2\}$. If z ends with B_2, then the derivation cannot be continued. If B_1 is the last letter of z, we can only apply P_4 and obtain $(Ab)^nC^r$ with $r < n$ (since we cancel at least the last letter of z). This process can be iterated, in each step we add a letter b after each A, and cancel at least one C. Finally, we get $(Ab^m)^n$ with $m \leq n$ (m gives the number of iterations, for which $1 \leq m \leq n$ holds). Now, by the use of P_5 we get $(ab^m)^n$ with $n \geq 1$ and $1 \leq m \leq n$. Thus

$$L(H) = \{(ab^m)^n \mid 1 \leq m \leq n\}.$$

We note that it is well-known that $L(H)$ cannot be generated by an ET0L system.

In this paper, we study the generative power of conditional ET0L systems, if one restricts to a class of subregular languages. For $X \in \mathcal{G}$, we define $\mathcal{CEL}(X)$ and $\mathcal{CEPL}(X)$ as the families of all languages which can be generated by conditional ET0L and conditional propagating ET0L system $(V, T, (P_1, R_1), \ldots, (P_n, R_n), w)$, where all languages R_i, $1 \leq i \leq n$, are in X.

By these definitions, the language from Example 1 is in $\mathcal{CEL}(COMB)$.

The following relations follow immediately from the definitions.

Lemma 1. *i) For all $X, Y \in \mathcal{G}$ with $X \subseteq Y$,*

$$\mathcal{CEL}(X) \subseteq \mathcal{CEL}(Y), \mathcal{CEPL}(X) \subseteq \mathcal{CEPL}(Y), \text{ and } \mathcal{CEPL}(X) \subseteq \mathcal{CEL}(X).$$

3 Some Equalities and Inclusions

In this section we prove inclusions $\mathcal{CEL}(X) \subseteq \mathcal{CEL}(Y)$ ($\mathcal{CEPL}(X) \subseteq \mathcal{CEPL}(Y)$) and equalities $\mathcal{CEL}(X') = \mathcal{CEL}(Y')$ ($\mathcal{CEPL}(X') = \mathcal{CEPL}(Y')$) for some families X, Y, X', and Y', respectively.

Lemma 2. *$\mathcal{CEL}(REG) = \mathcal{CEL}(UF)$ and $\mathcal{CEPL}(REG) = \mathcal{CEPL}(UF)$.*

Proof. It is known that any regular language is a union of finitely many union-free languages. Let

$$G = (V,\ T,\ (P_1, R_1),\ (P_2, R_2), \ldots,\ (P_n, R_n),\ \omega)$$

be a conditional ET0L system with regular conditions. Moreover, for $1 \leq i \leq n$, let

$$R_i = R_{i,1} \cup R_{i,2} \cup \cdots \cup R_{i,r_i},$$

where $R_{i,j}$ is union-free for $1 \leq j \leq n$. It is easy to prove that the ET0L system

$$(V, T, (P_1, R_{1,1}), \ldots, (P_1, R_{1,r_1}), (P_2, R_{2,1}), \ldots, (P_n, R_{n,1}), \ldots, (P_n, R_{n,r_n}), \omega)$$

with union-free conditions generates $L(G)$. Hence, $\mathcal{CL}(REG) \subseteq \mathcal{CL}(UF)$.

The converse inclusion follows by Lemma 1 and the inclusions given in the diagram of Figure 1.

Thus $\mathcal{CL}(REG) = \mathcal{CL}(UF)$.

For propagating ET0L systems, we have to repeat the proof. □

Lemma 3. *$\mathcal{CEL}(REG) \subseteq RE$ and $\mathcal{CEPL}(REG) \subseteq CS$.*

Proof. Let $L \in \mathcal{CEL}(REG)$, and let $G = (V, T, (P_1, R_1)(P_2, R_2), \ldots, (P_n, R_n), \omega)$ be a conditional ET0L system with regular conditions generating L. Then we construct a Turing machine M which works as follows (the detailed description of M is left to the reader):

(1) M checks whether ω is the word on the tape. If this is the case, M accepts; otherwise, it continues with (2).
(2) M chooses an i, $1 \leq i \leq n$, remembers i in the state, and chooses a decomposition $w = w_1 w_2 \ldots w_m$ of the tape content w; this can be done by writing $w_1 \# w_2 \# \ldots w_{m-1} \# w_m$ at the tape.
(3) M replaces each w_j $1 \leq j \leq m$, by some a_j where $a_j \to w_j \in P_i$ (if w_j is the empty word, this means that a_j with $a_j \to \lambda \in P_i$ is inserted) and cancels all symbols $\#$. If M can perform this step (i. e., the tape content $w_1 w_2 \ldots w_m$ is changed to $a_1 a_2 \ldots a_m$), it continues with (4); otherwise, M stops without accepting.
(4) M checks whether $a_1 a_2 \ldots a_m \in R_i$. In the affirmative case, M continues with (1); otherwise, M stops without accepting.

It is easy to see that a word w is accepted by M if and only if

$$w \vdash_{i_1} w_1 \vdash_{i_2} w_2 \vdash_{i_3} \cdots \vdash_{i_{q-1}} w_{q-1} \vdash_{i_q} \omega,$$

where $\vdash_i$ stands for applying (1) to (4) with i chosen in (2), if and only if

$$\omega \Longrightarrow_{P_{i_q}} w_{q-1} \Longrightarrow_{P_{i_{q-1}}} w_{q-2} \Longrightarrow_{P_{i_{q-2}}} \ldots \Longrightarrow_{P_{i_2}} w_1 \Longrightarrow_{P_{i_1}} w$$

$\omega \in R_{i_q}$, $w_k \in R_{i_k}$ for $1 \leq k \leq q-1$, and $w \in T^*$ if and only if $w \in L(G)$. Therefore the language accepted by M is $L(G)$. Thus $\mathcal{CEL}(REG) \subseteq RE$.

If the conditional T0L system G is propagating, each w_j of the decomposition is non-empty. Then it follows easily that the maximal length of the tape contents is twice the length of the input word. Hence the Turing machine is a linearly bounded automaton, which implies $\mathcal{CEPL}(REG) \subseteq CS$. □

Lemma 4. *$RE \subseteq \mathcal{CEL}(LOC_1)$.*

Proof. Let $L \in RE$. Then there is a grammar $G = (N, T, P, S)$ in Kuroda normal form (i. e., all rules have one of the following forms: $A \to BC$, $A \to a$, $A \to \lambda$, or $AB \to CD$ with $A, B, C, D \in N$ and $a \in T$) which generates L. Let P_1 be the set of all rules of P of the form $AB \to CD$ and $P_2 = P \setminus P_1$.

Let S'' and $\#$ be additional symbols not in $N \cup T$. We set

$$\begin{aligned} V' &= \{a' \mid a \in N \cup T\} \cup \{\#\}, \\ V_p &= \{a_p \mid a \in N \cup T\} \text{ for } p \in P_2, \\ V_p &= \{a_p \mid a \in N \cup T\} \cup \{a'_p \mid a \in N \cup T\} \text{ for } p \in P_1, \\ V_r &= \{a_r \mid a \in N \cup T \cup \{\#\}\} \cup \{a'_r \mid a \in N \cup T \cup \{\#\}\}, \\ V &= \{S''\} \cup N \cup T \cup V' \cup V_r \cup \bigcup_{p \in P} V_p. \end{aligned}$$

We now construct a conditional ET0L system H as follows: The basic and terminal alphabet are V and T, and the axiom is S''. Now we give all tables and conditions (if no rule is mentioned for some letter a, then $a \to a$ is the only rule for a in the production set):

$$(P_1, R_1) = (\{S'' \to \#S'\}, \{S''\}^+)$$

(we introduce from the axiom the word $\#S'$; the symbol $\#$ remembers the beginning of the word, because we shall use circular versions of a word; S' is the primed version of the start symbol of G),

$$\begin{aligned} (P_2, R_2) = (& \bigcup_{a \in N \cup T} \{a' \to a', a' \to a_r, a' \to a'_r\} \cup \bigcup_{a \in N \cup T} \bigcup_{p \in P_1} \{a' \to a_p\} \\ & \cup \bigcup_{a \in N \cup T} \bigcup_{p \in P_2} \{a' \to a_p, a' \to a'_p\} \cup \{\# \to \#, \# \to \#_r, \# \to \#'_r\}, \\ & V'(V')^+), \end{aligned}$$

(given a word over V', we can change some letters a' to their versions a_r and a'_r or their versions associated with a rule p; looking at the conditions of the tables defined below, the obtained word can only be handled if the changes are only done for the last letter or last and first letters and the introduced versions have to fit; more precise, from $x'w'y'$ with $x', y' \in V'$ and $w' \in (V')^*$, we can derive only $x'w'y_p$ with $p \in P_1$, or $x_pw'y'_p$ with $p \in P_2$, or $x_rw'y'_r$),

$$(P_{r,a,b}, R_{r,a,b}) = (\{a_r \to b'a', b'_r \to \lambda\}, \{a_r\}(V')^*\{b'_r\}) \text{ for } a, b \in V'$$

(if we obtained $a_r w' b'_r$ from $a'w'b'$, we now derive $b'a'w'$, i. e., we have performed a rotation step from $a'wb'$ to $b'a'w'$),

$$(P_p, R_p) = (\{A_p \to B'C'\}, (V')^+\{A_p\}) \text{ for } p = A \to BC$$

(if we obtained $x'w'A_p$, then we obtain $x'w'B'C'$, i. e., we have simulated an application $A \to BC$)

$$(P_p, R_p) = (\{A_p \to a'\}, (V')^+\{A_p\}) \text{ for } p = A \to a$$

(if we obtained $x'w'A_p$, then we obtain $x'w'a'$, i. e., we have simulated an application $A \to a$)

$$(P_p, R_p) = (\{A_p \to \lambda\}, (V')^+\{A_p\}) \text{ for } p = A \to \lambda$$

(if we obtained $x'w'A_p$, then we obtain $x'w'$, i. e., we have simulated an application $A \to \lambda$)

$$(P_p, R_p) = (\{B_p \to D', A'_p \to C'\}, \{B_p\}(V')^+\{A'_p\}) \text{ for } p = AB \to CD$$

(if we obtained $B_p w' A_p$, then we obtain $D'w'C'$, i. e., we have simulated an application $AB \to CD$ up to some rotation),

$$(P_3, R_3) = (\{a' \to a \mid a \in T\} \cup \{\# \to \lambda\}, \{\#\}T^*)$$

(if we have a word $\#x'$ with $x \in T$, then we can derive x).

We now prove that $L(G) \subseteq L(H)$. The basic idea is to start with $\#S'$ (produced by one application of P_1 to the axiom S''), perform circular shifts on a sentential form getting words of the form $x'_{r+1}x'_{r+2}\dots x'_n\#x'_1x'_2\dots x'_r$ and simulate the application of a rule in G by applying some table which only changes the last (if the rule is in P_1) or first and last letter (which are neighbouring letters in the non-rotated word, if the rule is from P_2), and to finish by a cancellation of # and returning to non-primed letters. Thus we can generate in H any word $w \in T^*$ which can be generated by G.

The converse inclusion $L(H) \subseteq L(G)$ holds, since we can perform only the rotation steps, or simulations of rules of P, or a cancellation of the primes, if we have a terminal word.

Since all the conditions of H are in LOC_1, the statement follows. □

Lemma 5. *$CS \subseteq \mathcal{CEPL}(LOC_2)$.*

Proof. Let $L \in CS$. Then there is a context-sensitive grammar $G = (N, T, P, S)$ in Kuroda normal form, i. e., all rules have the form $A \to B$, $A \to BC$, $AB \to CD$, and $A \to a$ with $A, B, C, D \in N$ and $a \in T$, such that $L = L(G)$. Let $p_1, p_2, \dots, p_r$ be the rules of P which have the form third mentioned form. For each rule $p_i = A_iB_i \to C_iD_i$, we introduce new letters A'_i and B'_i such that $A'_i \neq B'_j$ for $1 \leq i, j \leq r$ and $A'_i \neq A'_j$ and $B'_i \neq B'_j$ for $1 \leq i, j \leq r$, $i \neq j$. Let

$$V' = \{A'_i \mid 1 \leq i \leq n\} \cup \{B'_i \mid 1 \leq i \leq n\}.$$

Then we define the conditional ET0L system

$$H = (N \cup T \cup V',\ T,\ (Q, R),\ (Q', R'),\ S)$$

with

$$\begin{aligned}
Q =& \{X \to X \mid X \in N \cup T \cup V'\} \cup \{A \to w \mid A \to w \in P\} \\
& \cup \{A_i \to A_i' \mid p_i = A_iB_i \to C_iD_i, 1 \le i \le n\} \\
& \cup \{B_i \to B_i' \mid p_i = A_iB_i \to C_iD_i, 1 \le i \le n\} \\
R =& (N \cup T)^* \\
Q' =& \{X \to X \mid X \in N \cup T\} \cup \bigcup_{i=1}^{r} \{A_i' \to C_i, B_i' \to D_i\} \\
R' =& (N \cup T \cup \bigcup_{i=1}^{r} \{A_i'B_i'\})^*.
\end{aligned}$$

It is easy to see that the conditions R and R' belong to LOC_2.

We now prove that $L(H) = L(G)$.

Let $u \in (N \cup T)^+$ be a sentential form of G. Let $u \Longrightarrow v$ using some rule r of P which is different from all p_i, $1 \le i \le r$. Then $u = u_1Au_2$, $y = u_1wu_2$, and $r = A \to w$. This derivation can be simulated by a derivation according to table Q using $X \to X$ for all letters in u_1 and u_2 and $A \to w$ for A in the special position. If a rule $r_i = A_iB_i \to C_iD_i$ is applied to u we get $u = v_1A_iB_iv_2 \Longrightarrow v_1C_iD_iv_2 = y$. This derivation can be simulated in a two-step derivation

$$u = v_1A_iB_iv_2 \Longrightarrow_Q v_1A_i'B_i'v_2 \Longrightarrow_{Q'} v_1C_iD_iv_2 = y$$

where $X \to X$ from Q and Q' are applied to the letters of v_1 and v_2. Since G as well as H start with the axiom S, it is clear that $L(G) \subseteq L(H)$.

Assume that $x \in (N \cup T)^+$ is a sentential form of H. Then the application of Q' does not change x. Thus we have to apply Q. Let $x \Longrightarrow_Q y$. If y contains a letter A_i, then its successor in y is B_i since we cannot continue the derivation, otherwise (by the definition of R'). Let us assume without loss of generality (only the positions of the letters X_i, $1 \le i \le n$, and the subwords $A_{i_j}B_{i_j}$, $1 \le j \le m$, can occur in another order) that

$$x = u_1X_1u_2X_2u_3X_3 \ldots u_nX_nv_1A_{i_1}B_{i_1}v_2A_{i_2}B_{i_2}v_3 \ldots v_mA_{i_m}B_{i_m}v_{m+1}.$$

If $X \to X$ is applied to all letters of the words u_i, $1 \le i \le n$, and v_j, $1 \le j \le m+1$, $X_i \to w_i \in P$ is applied to all letters X_i, $1 \le i \le n$, and $A_{i_j} \to A_{i_j}'$ and $B_{i_j} \to B_{i_j}'$ are applied to the letters A_{i_j} and B_{i_j}, $1 \le j \le m$, we get

$$x \Longrightarrow_Q u_1w_1u_2w_2u_3w_3 \ldots u_nw_nv_1A_{i_1}'B_{i_1}'v_2A_{i_2}'B_{i_2}'v_3 \ldots v_mA_{i_m}'B_{i_m}'v_{m+1} = \overline{y}.$$

If $n \ge 1$ and $m \ge 1$, we have to apply Q' and get

$$\overline{y} \Longrightarrow_{Q'} u_1w_1u_2w_2u_3w_3 \ldots u_nw_nv_1C_{i_1}D_{i_1}v_2C_{i_2}D_{i_2}v_3 \ldots v_mC_{i_m}D_{i_m}v_{m+1} = y.$$

Since we have the derivation

$$\begin{aligned}
x &= u_1X_1u_2X_2u_3X_3\dots u_nX_nv_1A_{i_1}B_{i_1}v_2A_{i_2}B_{i_2}v_3A_{i_3}B_{i_3}v_4\dots v_mA_{i_m}B_{i_m}v_{m+1}\\
&\Longrightarrow u_1w_1u_2X_2u_3X_3\dots u_nX_nv_1A_{i_1}B_{i_1}v_2A_{i_2}B_{i_2}v_3A_{i_3}B_{i_3}v_4\dots v_mA_{i_m}B_{i_m}v_{m+1}\\
&\Longrightarrow u_1w_1u_2w_2u_3X_3\dots u_nX_nv_1A_{i_1}B_{i_1}v_2A_{i_2}B_{i_2}v_3A_{i_3}B_{i_3}v_4\dots v_mA_{i_m}B_{i_m}v_{m+1}\\
&\quad\dots\\
&\Longrightarrow u_1w_1u_2w_2u_3w_3\dots u_nw_nv_1A_{i_1}B_{i_1}v_2A_{i_2}B_{i_2}v_3A_{i_3}B_{i_3}v_4\dots v_mA_{i_m}B_{i_m}v_{m+1}\\
&\Longrightarrow u_1w_1u_2w_2u_3w_3\dots u_nw_nv_1C_{i_1}D_{i_1}v_2A_{i_2}B_{i_2}v_3A_{i_3}B_{i_3}v_4\dots v_mA_{i_m}B_{i_m}v_{m+1}\\
&\Longrightarrow u_1w_1u_2w_2u_3w_3\dots u_nw_nv_1C_{i_1}D_{i_1}v_2C_{i_2}D_{i_2}v_3A_{i_3}B_{i_3}v_4\dots v_mA_{i_m}B_{i_m}v_{m+1}\\
&\quad\dots\\
&\Longrightarrow u_1w_1u_2w_2u_3w_3\dots u_nw_nv_1C_{i_1}D_{i_1}v_2C_{i_2}D_{i_2}v_3\dots v_mC_{i_m}D_{i_m}v_{m+1}=y,
\end{aligned}$$

the derivation $x \Longrightarrow^* y$ in H can be simulated in G. If $n = 0$ or $m = 0$, we get analogously simulations. Thus $L(H) \subseteq L(G)$, too. □

Corollary 1. *$\mathcal{CEL}(REG) = RE$ and $\mathcal{CEPL}(REG) = CS$.*

Proof. By Lemmas 1, 4, 5, and 3,

$$RE \subseteq \mathcal{CEL}(LOC_1) \subseteq \mathcal{CEL}(REG) \subseteq RE$$

and

$$CS \subseteq \mathcal{CEPL}(LOC_2) \subseteq \mathcal{CEPL}(REG) \subseteq CS,$$

from which the statement immediately follows. □

Lemma 6. *$RE = \mathcal{CEL}(SUF)$ and $CS = \mathcal{CEPL}(SUF)$.*

Proof. i) Let $L \in RE$. Then, by Corollary 1, $L = L(G)$ for some ET0L system

$$G = (V, T, (P_1, R_1), (P_2, R_2), \dots (P_n, R_n), \omega)$$

with regular conditions. Let $V' = \{a' \mid a \in V\}$, and let S, F, and $\#$ be additional symbols. Then we set

$$\begin{aligned}
P_{init} &= \{S \to \#\omega'\} \cup \{a' \to a' \mid a' \in V' \cup \{\#, F\}\} \cup \{a \to F \mid a \in V\}\\
&\quad \text{and } R_{init} = \{S, \lambda\},\\
\overline{P_i} &= \{a' \to w' \mid a \to w \in P_i, a \in V\} \cup \{a \to a \mid a \in \{S, \#, F\}\} \cup \{a \to F \mid a \in V\}\\
&\quad \text{and } \overline{R_i} = \mathrm{Suf}(\{\#z' \mid z \in R_i\}) \text{ for } 1 \le i \le n,\\
P_{fin} &= \{\# \to \lambda\} \cup \{a' \to a \mid a \in T\} \cup \{a' \to F \mid a' \in (V' \setminus T') \cup V \cup \{S, F\}\}\\
&\quad \text{and } R_{fin} = \mathrm{Suf}(\{\#\}T^*)
\end{aligned}$$

and consider the conditional ET0L system

$$H = (V \cup V' \cup \{S, F, \#\}, T, (P_{init}, R_{init}), (\overline{P_1}, \overline{R_1}), \dots (\overline{P_n}, \overline{R_n}), (P_{fin}, R_{fin}), S).$$

Any derivation in H starts with $S \Longrightarrow \#\omega$ and in the sequel P_{init} cannot be applied. Moreover, by the definition of the production sets of H, any derivable word – except S – has the form $\#z'$ for some $z \in V^*$ or z with $z \in T^*$ or it contains at least one letter F. A set $\overline{P_i}$ is applicable to $\#z'$ if and only if $z \in R_i$, and its application yields $\#u'$ if and only if $z \Longrightarrow_{P_i} u$ holds in G. Furthermore, $\#z' \Longrightarrow z$ if and only if $z \in T^*$ by application of R_{fin}. From elements of $z \in T^*$ we obtain a word consisting only of Fs. If a word x contains an occurrence of F, then all words derivable from x contain an F, too; hence we cannot terminate the derivation. Now it follows easily that $L = L(H)$. Thus we have $RE \subseteq \mathcal{CEL}(SUF)$.

The converse inclusion follows from the relation $\mathcal{CEL}(SUF) \subseteq \mathcal{CEL}(REG) = RE$ by Lemma 1 and Corollary 1.

ii) Let $L \in CS$ and $T = \text{alph}(L)$. Moreover,

$$L = \bigcup_{a \in T} \{a\} L_a \text{ where } L_a = \{w \mid aw \in L_2\}.$$

Let

$$\begin{aligned}
T_1 &= \{a \mid a \in T, L_a = \emptyset\},\\
T_2 &= \{a \mid a \in T, L_a = \{\lambda\}\},\\
T_3 &= \{a \mid a \in T, \lambda \in L_a, w \in L_a \text{ for some non-empty word}\},\\
T_4 &= \{a \mid a \in T, \lambda \notin L_a, w \in L_a \text{ for some non-empty word}\}.
\end{aligned}$$

If $a \in T_3$, then we set $L'_a = L_a \setminus \{\lambda\}$. Then we get

$$L = T_2 \cup T_3 \cup \bigcup_{a \in T_3} \{a\} L'_a \cup \bigcup_{a \in T_4} \{a\} L_a.$$

By the closure properties of CS, L_a for all $a \in T_4$ and L'_a for $a \in T_3$ are context-sensitive languages and only consist of non-empty words. Hence, by Corollary 1, for any $a \in T_4$, there is a propagating conditional ET0L system G_a such that $L(G_a) = L_a$.

Now, for each $a \in T_4$, we construct the ET0L system G'_a with suffix-closed conditions as in the proof of the first statement of this lemma where we only change $\# \to \lambda$ to $\# \to a$ in the set P_{fin}. Then it follows as above that $L(G_a) = \{a\} L_a$ and G_a is propagating. Analogously, we can construct a propagating ET0L system G'_a for $a \in T_3$ such that $L(G'_a) = \{a\} L'_a$.

Now we rename all nonterminals in the ET0L systems G'_a, $a \in T_3 \cup T_4$ such that no nonterminal occurs in two different systems. Moreover, we change the rules and regular sets according to the renaming and add to each table rules $A \to A$ for all nonterminals not occurring in this table. For $a \in T_3 \cup T_4$, let

$$G''_a = (V', T, (P''_{1,a}, R''_{1,a}), (P''_{2,a}, R''_{2,a}), \ldots, (P''_{n_a,a}, R''_{n_a,a}), S_a).$$

Now we construct the propagating conditional ETOL system G with the alphabets $V' \cup \{S\}$ and T, where S is an additional symbol, the axiom S, the tables

$(P''_{i,a} \cup \{S \to S\}, R_{i,a})$ for $a \in T_3 \cup T_4$ and $1 \le i \le n_a$ and the additional table

$$(\{S \to a \mid a \in T_2 \cup T_3\} \cup \{S \to S_a \mid a \in T_3 \cup T_4\}, \{S, \lambda\}).$$

Obviously, G is propagating, all conditions of G are suffix-closed, and

$$\begin{aligned} L(G) &= T_2 \cup T_3 \cup \bigcup_{a \in T_3 \cup T_4} L(G''_a) = T_2 \cup T_3 \cup \bigcup_{a \in T_3 \cup T_4} L(G'_a) \\ &= T_2 \cup T_3 \cup \bigcup_{a \in T_3} \{a\} L'_a \cup \bigcup_{a \in T_4} \{a\} L_a = L. \end{aligned}$$

□

Lemma 7. *$RE = \mathcal{CEL}(CIRC)$ and $CS = \mathcal{CEPL}(CIRC)$.*

Proof. Let $L \in RE$. Then, by Corollary 1, $L = L(G)$ for some ET0L system

$$G = (V, T, (P_1, R_1), (P_2, R_2), \dots (P_n, R_n), \omega)$$

with regular conditions. From G we construct the conditional ET0L system H as in the first part of the proof of Lemma 6, where we take Circ instead of Suf in all cases. Then the obtained system has circular conditions. Moreover, $L(H) = L(G) = L$ can be shown as in the proof of Lemma 6. Thus we have $RE \subseteq \mathcal{CEL}(CIRC)$.

The converse inclusion follows from Lemma 3.

The proof of the second statement of the Lemma can be given by modifications analogous to those in the proof of the second statement of Lemma 6. □

Lemma 8. *$\mathcal{CEPL}(LOC_1) \subseteq EPT0L$.*

Proof. Let L be a language in $\mathcal{CEPL}(LOC_1)$. Then L is generated by some conditional ET0L system $G = (V, T, (P_1, R_1), (P_2, R_2), \dots, (P_n, R_n), \omega)$ with conditions in LOC_1. Then, for $1 \le i \le n$, $R_i = A_i B_i^* C_i \cup (A_i \cap C_i)$ for some sets $A_i, B_i, C_i \subseteq V$.

We first discuss the case that $\omega = azb$ for some $a, b \in V$ and $z \in V^*$.

Let

$$V' = \{a' \mid a \in V\},\ V'' = \{a'' \mid a \in V\} \text{ and } V''' = \{a''' \mid a \in V\}.$$

Moreover, for a set $U \subset V$, we set

$$U' = \{a' \mid a \in U\},\ U'' = \{a'' \mid a \in U\} \text{ and } U''' = \{a''' \mid a \in U\}.$$

For a word $w = a_1 a_2 \dots a_m$ with $a_i \in V$, we set $w' = a'_1 a'_2 \dots a'_m$. We define the EPT0L system

$$H = (V \cup V' \cup V'' \cup \{F\}, T, P'_1, P'_2, \dots, P'_n, Q, a''z'b''),$$

where

$$\begin{aligned} P'_i = {} & \{x'' \to y''v' \mid x \in A_i, x \to yv \in P_i\} \cup \{x' \to v' \mid x \in B_i, x \to v \in P_i\} \\ & \cup \{x''' \to v'y''' \mid x \in C_i, x \to vy \in P_i\} \\ & \cup \{x \to F \mid x \in (V' \cup V'' \cup V''' \cup \{F\}) \setminus (A''_i \cup B'_i \cup C'''_i)\} \end{aligned}$$

for $1 \leq i \leq n$ and

$$Q = \{x' \to x \mid x \in T\} \cup \{x'' \to x \mid x \in T\} \cup \{x''' \to x \mid x \in T\}$$
$$\cup \{x \to F \mid x \in (V' \cup V'' \cup V''' \cup \{F\}) \setminus (T' \cup T'' \cup T''')\}.$$

By these settings, without introducing F in a sentential form of the system H, $x_1vx_2 \Longrightarrow_{P_i} x_3ux_4$ in G if and only if $x_1''v'x_2''' \Longrightarrow_{P_i'} x_3''u'x_4'''$ in H and, moreover, $x_1''v'x_2''' \Longrightarrow_Q x_1vx_2$ in H if and only if $x_1vx_2 \in T^+$. Furthermore, if a letter F occurs in a sentential form w of H, then it also occurs in all sentential forms derivable from w in H. Thus it is obvious that

$$\omega = azb \Longrightarrow_{P_{i_1}} a_1z_1b_1 \Longrightarrow_{P_{i_2}} a_2z_2b_2 \Longrightarrow_{P_{i_3}} \ldots \Longrightarrow_{P_{i_k}} a_kz_kb_k$$

in G for some letters $a_i, b_i \in V$ and some words $z_i \in V^*$ for $1 \leq i \leq k$ if and only if

$$a''z'b''' \Longrightarrow_{P_{i_1}'} a_1''z_1'b_1''' \Longrightarrow_{P_{i_2}'} a_2''z_2'b_2''' \Longrightarrow_{P_{i_3}'} \ldots \Longrightarrow_{P_{i_k}'} a_k''z_k'b_k''' \Longrightarrow_Q a_kz_kb_k$$

in H. Therefore $L(G) = L(H)$ and it is shown that $L \in EPT0L$.

Now we discuss the case that ω is a letter. Then we define L_1 as the set of all letters, i. e., words of length 1, which can be derived in G, and $L_{2,i}$ with $1 \leq i \leq n$ as the set of all words of length ≥ 2, which can be obtained from $x \in L_1 \cap A_i \cap C_i$ by the application of a rule of P_i. Now we add a further letter S to the basic alphabet of H and a further table

$$Q' = \{S \to x \mid x \in L_1\} \cup \{S \to x \mid x \in L_{2,i}, 1 \leq i \leq n\}$$
$$\cup \{x \to x \mid x \in V \cup V' \cup V'' \cup V''' \cup \{F\}\}.$$

Now it follows analogously to the above considerations that $L(G) = L(H)$ holds. □

Lemma 9. $\mathcal{CEPL}(DEF) \subseteq EPT0L$.

Proof. Let $R = A \cup V^*B$ with finite sets $A \subseteq V^*$ and $B \subseteq V^*$. Let m be a number which is greater than the maximal length of words in A and B. Then we have

$$R = \{w \mid |w| \leq m, w \in L\} \cup V^*(\bigcup_{w \in B} V^{m-|v|}\{w\}),$$

i. e., R can be represented as $R = A' \cup V^*B'$ with $A' \subseteq \bigcup_{j=1}^{m} V^j$ and $B' \subseteq V^m$.

Let $L \in \mathcal{CEPL}(DEF)$. Then $L = L(G)$ for some propagating ET0L system

$$G = (V, T, (P_1, R_1), (P_2, R_2), \ldots, (P_n, R_n), \omega)$$

with definite conditions. By the above observation, without loss of generality, we can assume that there is a number $m \geq |\omega|$ such that, for $1 \leq i \leq n$, $R_i = A_i \cup V_i^*B_i$ with $A_i \subseteq \bigcup_{j=1}^{m} V_i^j$ and $B_i \subseteq V_i^m$ for some $V_i \subseteq V$.

Moreover, let $V' = \{a' \mid a \in V\}$ and $V'' = \{[w] \mid w \in V^*, |w| \leq m\}$. We construct the EPT0L system

$$H = (V \cup V' \cup V''), T, \overline{P_1}, \overline{P_2}, \ldots, \overline{P_n}, Q, [\omega])$$

with

$$\begin{aligned}\overline{P_i} =&\{a \to a \mid a \in V \cup \{F\}\} \cup \{a' \to z' \mid a \in V_i, a \to z \in P_i\}\\ &\cup \{[w] \to [z] \mid [w] \in V'', w \in A_i \cup B_i, w \Longrightarrow_{P_i} z, |z| \leq m\}\\ &\cup \{[w] \to x'[z] \mid [w] \in V'', w \in A_i \cup B_i, w \Longrightarrow_{P_i} xz, |z| = m\}\\ &\cup \{a' \to F \mid a \in V \setminus V_i\} \cup \{[w] \to F \mid [w] \in V'', w \notin A_i \cup B_i\}\end{aligned}$$

for $1 \leq i \leq n$ and

$$Q = \{a \to a \mid a \in V \cup \{F\}\} \cup \{a' \to a \mid a \in V\} \cup \{[w] \to w \mid [w] \in V''\}.$$

By the construction, all sentential forms have the form $[w]$, $x'[w]$ or x with $[w] \in V''$ and $x \in V^+$. Furthermore, we have the derivations $[w] \Longrightarrow_{\overline{P_i}} [z]$ if and only if $w \Longrightarrow_{P_i} z$, $[w] \Longrightarrow_{\overline{P_i}} x'[z]$ if and only if $w \Longrightarrow_{P_i} xz$. Moreover, if a word $y \in V^*$ is obtained by using $x'[w] \Longrightarrow_Q y$ (if $xw = y$ and $|y| \geq m+1$) or $[y] \Longrightarrow_Q y$ (if $|y| \leq m$), then it is not changed by further derivation steps, because we have only the rule $a \to a$ for $a \in V$ in all tables. Thus any derivation in H has the form

$$\begin{aligned}[\omega] &\Longrightarrow_{\overline{P_{i_1}}} [w_1] \Longrightarrow_{\overline{P_{i_2}}} \ldots \Longrightarrow_{\overline{P_{i_r}}} [w_r]\\ &\Longrightarrow_{\overline{P_{i_{r+1}}}} x'_1[w_{r+1}] \Longrightarrow_{\overline{P_{i_{r+2}}}} x'_2[w_{r+2}] \Longrightarrow_{\overline{P_{i_{r+3}}}} \ldots \Longrightarrow_{\overline{P_{i_{r+s}}}} x'_s[w_{r+s}]\\ &\Longrightarrow_Q x_s w_{r+s} \Longrightarrow x_s w_{r+s} \Longrightarrow \ldots\end{aligned}$$

with $[w_i] \in V''$ for $1 \leq i \leq r+s$ and $x_j \in V^*$ for $1 \leq j \leq s$; and such a derivation exists if and only there is a derivation

$$\begin{aligned}\omega &\Longrightarrow_{P_{i_1}} w_1 \Longrightarrow_{P_{i_2}} \ldots \Longrightarrow_{P_{i_r}} w_r \Longrightarrow_{P_{i_{r+1}}} x_1 w_{r+1}\\ &\Longrightarrow_{P_{i_{r+2}}} x_2 w_{r+2} \Longrightarrow_{P_{i_{r+3}}} \ldots \Longrightarrow_{P_{i_{r+s}}} x_s w_{r+s}\end{aligned}$$

in G exists. Therefore, $L(H) = L(G)$ and $L \in EPT0L$. □

Lemma 10. *$ET0L \subseteq \mathcal{CEPL}(MON)$.*

Proof. Let L be a language in *ET0L*. Then there is a propagating ET0L system $G = (V, T, P_1, P_2, \ldots, P_r, \omega)$ generating L. In an ET0L system, any table can be applied to any sentential form. Thus the conditional propagating ET0L system $(V, T, (P_1, V^*), (P_2, V^*), \ldots, (P_r, V^*), \omega)$ with monoidal conditions generates L, too. Therefore $ET0L \subseteq \mathcal{CEPL}(MON)$. □

Lemma 11. *$\mathcal{CEPL}(COMM) = MAT$ and $\mathcal{CEL}(COMM) = MAT^\lambda$*

Proof. i) $MAT^\lambda \subseteq \mathcal{CEL}(COMM)$.

We recall that any recursively enumerable language can be generated by a matrix grammar G in 2-normal form (see Lemma 1.2.3 in [4]), i. e., by a matrix grammar $G = (N_1 \cup N_2 \cup \{S\},\ T,\ M,\ S,\ Q)$ where all matrices of M have one of the following forms

– $(S \to AX)$ with $A \in N_1$ and $X \in N_2$,

– $(A \to w,\ X \to Y)$ with $A \in N_1$, $w \in (N_1 \cup T)^*$, and $X, Y \in N_2$,
– $(A \to w,\ X \to \lambda)$ with $A \in N_1$, $w \in (N_1 \cup T)^*$, and $X \in N_2$,

and Q contains only rules of the form $A \to w$.

Let L be a language in MAT^{λ}. Then L is generated by a matrix grammar $G = (N_1 \cup N_2 \cup \{S\},\ T,\ M,\ S,\ Q)$ which satisfies the above mentioned normal form conditions. Let $m_1, m_2, \ldots m_r$ be the matrices $(A \to w, X \to z)$ of M, $z \in N_2 \cup \{\lambda\}$ with $A \to w \notin F$ and $m_{r+1}, m_{r+2}, \ldots m_s$ be the matrices $(A \to w, X \to z)$ of M, $z \in N_2 \cup \{\lambda\}$ with $A \to w \in F$ We set

$$V = N_1 \cup N_2 \cup \{S\} \cup T \cup \{B' \mid B \in N_1 \cup N_2 \cup T\} \cup \bigcup_{(A \to w, X \to z) \in M} \{A_m, X_m\}$$

and

$$(P, R) = (\{S \to AX \mid (S \to AX) \in M\}, \{S\}).$$

With a matrix $m = (A \to w, X \to z)$ with $z \in N_2$ or $z = \lambda$, we associate $(P_{m,1}, R_{m,1})$ and $(P_{m,2}, R_{m,2})$ defined by

$$\begin{aligned} P_{m,1} &= \{A \to A, A \to A_m, X \to X_m\}, \\ R_{m,1} &= (N_1 \cup N_2 \cup T)^+, \\ P_{m,2} &= \{A_m \to w, X_m \to Y\}, \\ R_{m,2} &= \{w \mid w \in (N_1 \cup T \cup \{A_m, X_m\})^+, \#_{A_m} = \#_{X_m} = 1\}, \end{aligned}$$

and if $A \to w$ is an element of F, we add

$$(P'_{m,2}, R'_{m,2}) = (\{X_m \to Y\},\ \{w \mid w \in ((N_1 \setminus \{A\}) \cup T \cup \{X_m\})^+, \#_{X_m} = 1\}).$$

We construct the conditional Lindenmayer system

$$\begin{aligned} G' = (V,\ T,\ (P, R),\ & (P_{m_1,1}, R_{m_1,1}), (P_{m_1,2}, R_{m_1,2}), \\ & \ldots, (P_{m_r,1}, R_{m_r,1}), (P_{m_r,2}, R_{m_r,2}), \\ & (P_{m_{r+1},1}, R_{m_{r+1},1}), (P_{m_{r+1},2}, R_{m_{r+1},2}), (P'_{m_{r+1},2}, R'_{m_{r+1},2}), \\ & \ldots, (P_{m_s,1}, R_{m_s,1}), (P_{m_s,2}, R_{m_s,2}), (P'_{m_s,2}, R'_{m_s,2}),\ S). \end{aligned}$$

Obviously, all conditions are commutative. We now prove that $L(G') = L(G)$.

In both devices any derivation starts with $S \Longrightarrow AX$.

Now let $z_1 A z_2 X$ be a sentential form of G, and let $m = (A \to w, X \to Y)$ be a matrix of M. Then in G we get $z_1 w z_2 Y$. In G', by application of $(P_{m,1}, R_{m,1})$, we obtain a word which differs from $z'_1 A_m z'_2 X_m$ where z'_1 and z'_2 are obtained from z_1 and z_2 by replacing some As by A_m. However, the derivation can only continued if there are no A_ms in z'_1 and z'_2, i. e., we obtained $z_1 A_m z_2 X_m$. Now only $(P_{m,2}, R_{m,2})$ can be applied which yields $z_1 w z_2 Y$. Therefore we have simulated a derivation step of G. If $A \to w$ is in F and the sentential form zX does not contain a letter A, then we get in G the word zY, and in G' we have the simulation $zX \Longrightarrow zX_m \Longrightarrow zY$.

Obviously, a successful derivation in G' consists only of the mentioned derivation steps.

Moreover, the derivation in G stops if and only if no table $(P_{m,2}, R_{m,2})$ and $(P'_{m,2}, R'_{m,2})$ changes the sentential form.

Thus $L(G') = L(G)$ follows.

ii) $MAT \subseteq \mathcal{CEPL}(COMM)$.

This can be shown analogously. We have only to start with the accurate normal form (see Definition 1.3.2 and Lemma 1.3.7 in [4]).

iii) $\mathcal{CEL}(COMM) \subseteq MAT^{\lambda}$

Let $L \in \mathcal{CEL}(COMM)$. Then there is a conditional Lindenmayer system $G = (V, T, (P_1, R_1), (P_2, R_2, \ldots, (P_n, R_n), w)$ such that, for any i, $1 \leq i \leq n$, R_i is a commutative and regular language.

Let

$$V = \{A_1, A_2, \ldots, A_m\} \text{ and } T = \{A_{p+1}, A_{p+2}, \ldots, A_m\}.$$

Obviously, for $1 \leq i \leq n$, R_i is a set over V and $R_i = \pi_V^{-1}(M_i)$ for some semi-linear set M_i. Let

$$M_i = \bigcup_{j=1}^{r_i} M_{i,j}$$

with

$$\begin{aligned} M_{i,j} \quad = \quad & \{(a_{1,i,j}, a_{2,i,j}, \ldots, a_{m,i,j}) \\ & + \sum_{k=1}^{t_{i,j}} \alpha_k (b_{1,k,i,j}, b_{2,k,i,j}, \ldots, b_{m,k,i,j}) \mid \alpha_j \in \mathbb{N} \text{ for } 1 \leq k \leq t_{i,j}\} \end{aligned}$$

for some $a_{r,i,j}$ and $b_{r,k,i,j}$, $1 \leq r \leq m$, $1 \leq i \leq n$, $1 \leq j \leq r_i$, and $1 \leq k \leq t_{i_j}$.

We define the matrix grammar $G' = (N', T', M, S, Q)$ where

$$\begin{aligned} N' \quad &= \quad \{S, Z, \#\} \cup \bigcup_{i=1}^{m} \{A'_i, A''_i\} \cup \bigcup_{j=1}^{n} \{Z_i, Z'_i\} \cup \bigcup_{\substack{1 \leq i \leq n \\ 1 \leq j \leq t_i}} \{Z_{i,j}\} \\ T' \quad &= \quad T \cup \{X\}, \\ Q \quad &= \quad \{A'_i \to \# \mid 1 \leq i \leq m\} \cup \{A''_i \to \# \mid 1 \leq i \leq m\}, \end{aligned}$$

and M consists of all matrices constructed as follows. As initial rules we take all rules

$$(S \to Z_i w') \text{ for } 1 \leq i \leq n$$

(we generate a primed version of the axiom w of G accompanied by some control symbol Z_i).

For any $1 \leq i \leq n$, $1 \leq i' \leq n$, and $1 \leq j \leq r_i$, we introduce the matrices

$$\begin{aligned} & (Z_i \to Z_{i,j}, (A'_1 \to A''_1)^{a_{1,i,j}}, (A'_2 \to A''_2)^{a_{2,i,j}}, \ldots, (A'_m \to A''_m)^{a_{m,i,j}}) \\ & (Z_{i,j} \to Z_{i,j}, (A'_1 \to A''_1)^{b_{1,k,i,j}}, (A'_2 \to A''_2)^{b_{2,k,i,j}}, \ldots, (A'_m \to A''_m)^{b_{m,k,i,j}}) \\ & \qquad \text{for } 1 \leq k \leq t_{i,j} \\ & (Z_{i,j} \to Z'_i, A'_1 \to \#, A'_2 \to \#, \ldots, A'_m \to \#) \text{ for } 1 \leq j \leq r_i, \end{aligned}$$

(applying these matrices to a sentential form $Z_i v'$ for some $v' \in (V')^*$ one checks whether the Parikh vector of v is contained in $M_{i,j}$; thus the $Z_i' v''$ can only be obtained if the sentential form is contained in R_i)

$$(Z_i' \to Z_i', A'' \to w') \text{ for } A \to w \in P_i$$

(after checking that the sentential form is in R_i we apply the rules of P_i),

$$(Z_i' \to Z_{i'}, A_1'' \to \#, A_2'' \to \#, \ldots, A_n'' \to \#)$$

(if all letters of v'' are replaced, i. e., we get z' where $v \Longrightarrow z$ holds in G and have simulated a derivation step in G, we can start the same process with i'),

$$\begin{aligned}&(Z_i \to Z, A_1' \to \#, A_2' \to \#, \ldots, A_p' \to \#),\\ &(Z \to Z, A_q' \to A_q) \text{ for } p+1 \leq q \leq m,\\ &(Z \to X, A_{q+1}' \to \#, A_{q+2}' \to \#, \ldots, A_m' \to \#).\end{aligned}$$

(if $Z_i v'$ does not contain the letters $A_1', A_2', \ldots A_p'$, i. e., v is a word over the terminal alphabet T, we replace all letters A_q' by A_q, and finally Z by X).

By the given explanations, it is easy to see that $L(G') = \{X\}L(G)$.

Thus $\{X\}L(G) \in MAT^{\lambda}$. By the closure properties of MAT (see [4], page 48), $L(G) \in MAT^{\lambda}$ which proves the statement.

iv) $\mathcal{CEPL}(COMM) \subseteq MAT$

Since the construction in iii) produces no erasing rules in the matrix grammar if the conditional Lindenmayer system contains no erasing rules, the statement follows by the same construction. □

Lemma 12. $\mathcal{CEPL}(FIN) = \mathcal{CEL}(FIN) = FIN$

Proof. Obviously, any language in $\mathcal{CEL}(FIN)$ is finite. Thus $\mathcal{CEL}(FIN) \subseteq FIN$.

Let $L \subset T^+$ be finite language (note that by our setting that languages are equal if they differ at most in the empty word, we can ignore the empty word, if it is in L). It is easy to see that the propagating ET0L system

$$(\{S\} \cup T, T, (\{S \to w \mid w \in L\}, \{S\}), S)$$

with a finite condition generates L. Thus $FIN \subseteq \mathcal{CEPL}(FIN)$.

By these inclusions and Lemma 1, we get the statement of the lemma. □

4 Summary and Conclusions

By a combination of the lemmas above and Example 1, we get the following theorem.

Theorem 1. *For all $s \geq 1$ and $r \geq 2$, the diagram given in Figure 2 holds.*

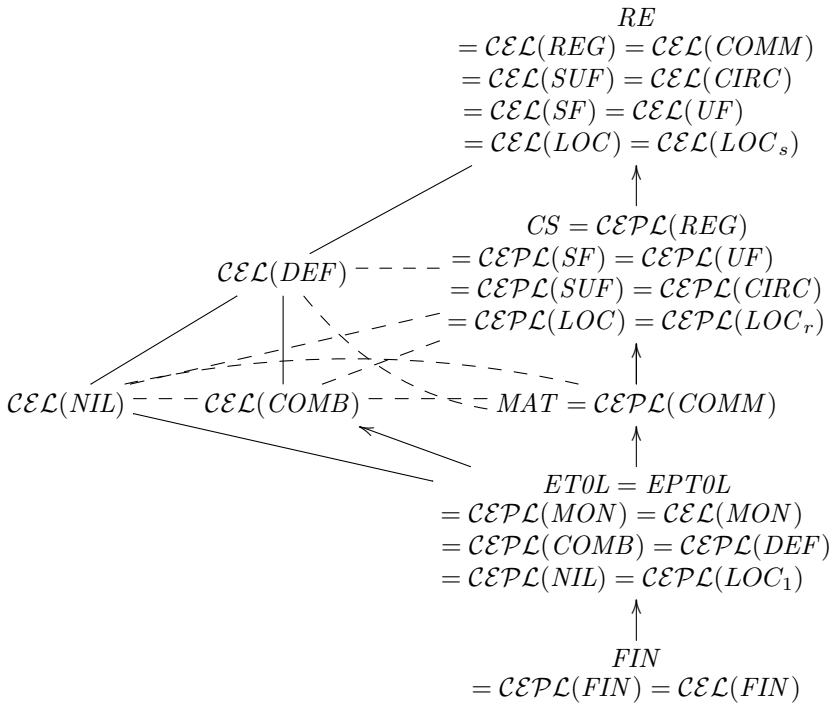

Figure 2: Hierarchy of language families $\mathcal{CEL}(X)$ and $\mathcal{CEPL}(X)$ with $X \in \mathcal{G}$ (an arrow from Z_1 to Z_2 denotes $Z_1 \subset Z_2$; a line from Z_1 to a higher positioned Z_2 stands for $Z_1 \subseteq Z_2$; the relation between families which are connected by a broken line is unknown; and if two families are not connected by a directed path or a broken line, then they are incomparable)

If one only considers the propagating families, then the hierarchy is completely determined. However, in the general case, there are some open problems related to the families $\mathcal{CEL}(NIL)$, $\mathcal{CEL}(COMB)$, and $\mathcal{CEL}(DEF)$; essentially we only have the relations which follow directly from the relation between the subregular families.

The obtained picture is very similar to that which was obtained for (sequential) context-free conditional grammars (for a definition see [4]). Especially,

$$\mathcal{CEL}(X) = Z \in \{RE, CS, MAT, ET0L\}$$

implies that the family of context-free conditional grammars with conditions from X coincides with Z, too; this implication also holds for systems/grammars with only non-erasing rules. However, the families of context-free conditional grammars with definite, nilpotent, and combinational conditions are also equal to $ET0L$. In contrast, for Lindenmayer systems $ET0L \subset \mathcal{CEL}(COMB) \subseteq \mathcal{CEL}(DEF)$.

References

[1] Dassow, J. Subregularly controlled derivations: the context-free case. *Rostocker Mathematisches Kolloquium*, **34**:61–70, 1988.

[2] Dassow, J., Grammars with commutative, circular, and locally testable conditions. In Ésik, Z., and Fülöp, Z., editors, *Automata, Formal Languages, and Related Topics – Dedicated to Ferenc Gécseg on the Occasion of his 70th Birthday*, pages 27–37. University of Szeged, 2009.

[3] Dassow, J. Subregular restrictions for some language generating devices. In Freund, R., Holzer, M., Truthe, B., and Ultes-Nitsche, U., editors, *NCMA 2012, Proceedings*, pages 11–26. books@ocg.at 290, Österreichische Computer Gesellschaft, Austria, 2012.

[4] Dassow J. and Păun, Gh. *Regulated Rrewriting in Formal Language Theory.* Springer-Verlag, Berlin, 1989.

[5] Dassow, J. and Rudolf, St. Conditional Lindenmayer systems with subregular conditions: the non-extended case. *RAIRO - Theor. Inf. Appl.*, **48**:127–147, 2014.

[6] Dassow, J., Stiebe, R., and Truthe, B. Generative capacity of subregularly tree controlled grammars. *International Journal of Foundations of Computer Science*, **21**:723–740, 2010.

[7] Gécseg, F. On some classes of tree automata and tree languages. *Annales Academiae Scientiarum Fennicae*, **25**: 325–336, 2000.

[8] Gécseg, F. and Gyurica, Gy. On the closedness of DR tree languages under Boolean operations. *Acta Cybernetica*, **17**:449–457, 2006.

[9] Gécseg, F. and Imreh, B. On isomorphic representations of generalized definite automata. *Acta Cybernetica*, **15**:33–43, 2001.

[10] Gécseg, F. and Imreh, B. on definite and nilpotent DR tree languages. *Journal of Automata, Languages, and Combinatorics*, **9**:55–60, 2004.

[11] Gécseg, F. and Peák, I. *Algebraic Theory of Automata.* Academiai kiado, Budapest, 1972.

[12] Havel, I. M. The theory of regular events II. *Kybernetika*, **5**:520–544, 1969.

[13] Martín-Vide, C. and Mitrana, V. Networks of Evolutionary Processors: Results and Perspectives. In: *Molecular Computational Models: Unconventional Approaches*, pages 78–114, 2005.

[14] Păun, Gh. *Marcus Contextual Grammars.* Kluwer Publ. House, Doordrecht, 1998.

[15] Rozenberg, G. and Salomaa, A. *The Mathematical Theory of L Systems*. Academic Press, New York, 1980.

[16] Rozenberg, G. and Salomaa, A., editors. *Handbook of Formal Languages*. Springer-Verlag, Berlin, 1997.

[17] Rozenberg, G. and von Solms, S. H. Priorities on context conditions in rewriting systems. *Inform. Sci.*, **14**:15–51, 1978.

[18] Shyr, H. J. *Free Monoids and Languages*. Hon Min Book Co., Taichung, Taiwan, 1991.

[19] Starke, P. H. *Abstrakte Automaten*. Deutscher Verlag der Wissenschaften, Berlin, 1969.

[20] Wiedemann, B. *Vergleich der Leistungsfähigkeit endlicher determinierter Automaten*. Diplomarbeit, Universität Rostock, 1978.

Received 13th May 2015

Acta Cybernetica **22** (2015) 335–347.

On a Property of Non Liouville Numbers*

Jean-Marie De Koninck† and Imre Kátai‡

Dedicated to the memory of Professor Ferenc Gécseg

Abstract

Let α be a non Liouville number and let $f(x) = \alpha x^r + a_{r-1}x^{r-1} + \cdots + a_1x + a_0 \in \mathbb{R}[x]$ be a polynomial of positive degree r. We consider the sequence $(y_n)_{n\geq 1}$ defined by $y_n = f(h(n))$, where h belongs to a certain family of arithmetic functions and show that $(y_n)_{n\geq 1}$ is uniformly distributed modulo 1.

Keywords: non Liouville numbers, uniform distribution modulo 1

1 Introduction and notation

Let $t(n)$ be an arithmetic function and let $f \in \mathbb{R}[x]$ be a polynomial. Under what conditions is the sequence $(f(t(n)))_{n\geq 1}$ uniformly distributed modulo 1 ? In the particular case where f is of degree one, the problem is partly solved. For instance, it is known that, if α is an irrational number and if $t(n) = \omega(n)$ or $\Omega(n)$, where $\omega(n)$ stands for the number of distinct prime factors of n and $\Omega(n)$ for the number of prime factors of n counting their multiplicity, with $\omega(1) = \Omega(1) = 0$, then the sequence $(\{\alpha t(n)\})_{n\geq 1}$ is uniformly distributed modulo 1 (here $\{y\}$ stands for the fractional part of y). In 2005, we [1] proved that if α is a positive irrational number such that for each real number $\kappa > 1$ there exists a positive constant $c = c(\kappa, \alpha)$ for which the inequality $\|\alpha q\| > c/q^\kappa$ holds for every positive integer q, then the sequence $(\{\alpha\sigma(n)\})_{n\geq 1}$ is uniformly distributed modulo 1. (Here $\|x\|$ stands for the distance between x and the nearest integer and $\sigma(n)$ stands for the sum of the positive divisors of n.) Observe that one can construct an irrational number α for which the corresponding sequence $(\{\alpha\sigma(n)\})_{n\geq 1}$ is not uniformly distributed modulo 1. On the other hand, given an integer $q \geq 2$ and letting $s_q(n)$ stand for the sum of the digits of n expressed in base q, it is not hard to prove that, if α is an irrational number, the sequence $(\{\alpha s_q(n)\})_{n\geq 1}$ is uniformly distributed modulo 1. In fact, in the past 15 years, important results have been obtained concerning the

*The research of the first author was supported in part by a grant from NSERC.

†Département de mathématiques et de statistique Université Laval, Québec G1V 0A6, Canada. E-mail: jmdk@mat.ulaval.ca

‡Computer Algebra Department, Eötvös Loránd University, 1117 Budapest, Pázmány Péter Sétány I/C, Hungary. E-mail: katai@inf.elte.hu

topic of the so-called q-ary arithmetic functions. For instance, it was proved that the sequence $(\{\alpha s_q(p)\})_{p\in\wp}$ (here $\wp$ is the set of all primes) is uniformly distributed modulo 1 if and only if $\alpha \in \mathbb{R}\setminus\mathbb{Q}$. In 2010, answering a problem raised by Gelfond [10] in 1968, Mauduit and Rivat [13] proved that the sequence $(\{\alpha s_q(n^2)\})_{n\ge 1}$ is uniformly distributed modulo 1 if and only if $\alpha \in \mathbb{R}\setminus\mathbb{Q}$.

Recall that an irrational number β is said to be a *Liouville number* if for all integers $m \ge 1$, there exist two integers t and $s > 1$ such that

$$0 < \left|\beta - \frac{t}{s}\right| < \frac{1}{s^m}.$$

Hence, Liouville numbers are those real numbers which can be approximated "quite closely" by rational numbers.

Here, if α is a non Liouville number and

$$f(x) = \alpha x^r + a_{r-1}x^{r-1} + \cdots + a_1 x + a_0 \in \mathbb{R}[x] \quad \text{is of degree } r \ge 1, \tag{1}$$

we prove that $(f(t(n)))_{n\ge 1}$ is uniformly distributed modulo 1, for those arithmetic functions $t(n)$ for which the corresponding function $a_{N,k} := \dfrac{1}{N}\#\{n \le N : t(n) = k\}$ is "close" to the normal distribution as N becomes large.

Given $\mathcal{P} \subseteq \wp$, let $\Omega_{\mathcal{P}}(n) = \sum_{\substack{p^r \| n \\ p\in\mathcal{P}}} r$. From here on, we let $q \ge 2$ stand for a fixed integer. Now, consider the sequence $(y_n)_{n\ge 1}$ defined by $y_n = f(h(n))$, where $h(n)$ is either one of the five functions

$$\omega(n), \qquad \Omega(n), \qquad \Omega_{\mathcal{P}}(n), \qquad s_q(n), \qquad s_q(n^2). \tag{2}$$

Here, we show that the sequence $(y_n)_{n\ge 1}$ is uniformly distributed modulo 1.

For the particular case $h(n) = s_q(n)$, we also examine an analogous problem, as n runs only through the primes. Finally, we consider a problem involving strongly normal numbers.

Recall that the *discrepancy* of a set of N real numbers $x_1, \ldots, x_N$ is the quantity

$$D(x_1, \ldots, x_N) := \sup_{[a,b)\subseteq[0,1)} \left| \frac{1}{N} \sum_{\{x_\nu\}\in[a,b)} 1 - (b-a) \right|.$$

For each positive integer N, let

$$M = M_N = \lfloor \delta_N \sqrt{N} \rfloor, \qquad \text{where } \delta_N \to 0 \text{ and } \delta_N \log N \to \infty \text{ as } N \to \infty. \tag{3}$$

We shall say that an infinite sequence of real numbers $(x_n)_{n\ge 1}$ is *strongly uniformly distributed* mod 1 if

$$D(x_{N+1}, \ldots, x_{N+M}) \to 0 \qquad \text{as} \quad N \to \infty$$

for every choice of M (and corresponding δ_N) satisfying (3). Then, given a fixed integer $q \ge 2$, we say that an irrational number α is a *strongly normal number*

in base q (or a strongly q-normal number) if the sequence $(x_n)_{n\geq 1}$, defined by $x_n = \{\alpha q^n\}$, is strongly uniformly distributed modulo 1. The concept of strong normality was recently introduced by De Koninck, Kátai and Phong [2].

We will at times be using the standard notation $e(x) := \exp\{2\pi i x\}$. Finally, we let φ stand for the Euler totient function.

2 Background results

The sum of digits function $s_q(n)$ in a given base $q \geq 2$ has been extensively studied over the past decades. Delange [4] was one of the first to study this function. Drmota and Rivat [7], [14] studied the function $s_q(n^2)$ and then, very recently, Drmota, Mauduit and Rivat [9] analyzed the distribution of the function $s_q(P(n))$, where $P \in \mathbb{Z}[x]$ is a polynomial of a certain type.

Here, we state as propositions some other results and recall two relevant results of Halász and Kátai.

First, given an integer $q \geq 2$, we set

$$\mu_q = \frac{q-1}{2}, \qquad \sigma_q^2 = \frac{q^2-1}{12}.$$

Proposition 1. *Let $\delta > 0$ be an arbitrary small number and let $\varepsilon > 0$. Then, uniformly for $\left|k - \mu_q \log_q N\right| < \frac{1}{\delta}\sqrt{\log_q N}$,*

$$\#\{n \leq N : s_q(n) = k\} = \frac{N}{\sqrt{2\pi\sigma_q^2 \log_q N}}\left(\exp\left\{-\frac{(k-\mu_q \log_q N)^2}{2\sigma_q^2 \log_q N}\right\} + O\left(\frac{1}{\log^{\frac{1}{2}-\varepsilon} N}\right)\right).$$

Proof. This result is in fact a particular case of Proposition 3 below. □

Proposition 2. *Let $\varepsilon > 0$. Uniformly for all integers $k \geq 0$ such that $(k, q-1) = 1$,*

$$\#\{p \leq N : s_q(p) = k\} = \frac{q-1}{\varphi(q-1)}\frac{\pi(N)}{\sqrt{2\pi\sigma_q^2 \log_q N}}\left(\exp\left\{-\frac{(k-\mu_q \log_q N)^2}{2\sigma_q^2 \log_q N}\right\} + O\left(\frac{1}{\log^{\frac{1}{2}-\varepsilon} N}\right)\right).$$

Proof. This is Theorem 1.1 in the paper of Drmota, Mauduit and Rivat [8]. □

Let $G = (G_j)_{j\geq 0}$ be a strictly increasing sequence of integers, with $G_0 = 1$. Then, each non negative integer n has a unique representation as $n = \sum_{j\geq 0} \epsilon_j(n) G_j$ with integers $\epsilon_j(n) \geq 0$ provided that $\sum_{j<k} \epsilon_j(n) G_j < G_k$ for all integers $k \geq 1$. Then, the sum of digits function $s_G(n)$ is given by

$$s_G(n) = \sum_{j\geq 0} \epsilon_j(n). \tag{4}$$

Setting $a_{N,k} := \#\{n \le N : s_G(n) = k\}$, consider the related sequence $(X_N)_{N\ge1}$ of random variables defined by

$$P(X_N = k) = \frac{a_{N,k}}{N},$$

so that the expected value of X_N and its variance are given by

$$E[X_N] = \frac{1}{N}\sum_{n\le N} s_G(n) \qquad \text{and} \qquad V[X_N] = \frac{1}{N}\sum_{n\le N}(s_G(n) - E[X_N])^2. \tag{5}$$

Let us choose the sequence $(G_j)_{j\ge0}$ as the particular sequence

$$G_0 = 1, \qquad G_j = \sum_{i=1}^{j} a_i G_{j-1} + 1 \quad (j > 0), \tag{6}$$

where the a_i's are simply the positive integers appearing in the Parry α-expansion (here $\alpha > 1$ is a real number) of 1, that is

$$1 = \frac{a_1}{\alpha} + \frac{a_2}{\alpha^2} + \frac{a_3}{\alpha^3} + \cdots$$

It can be shown (see Theorem 2.1 of Drmota and Gajdosik [5]) that, for such a sequence $(G_j)_{j\ge0}$, setting

$$G(z,u) := \sum_{j=1}^{\infty}\left(\sum_{\ell=0}^{a_j-1} z^\ell\right) z^{a_1+\cdots+a_{j-1}} u^j$$

and letting $1/\alpha(z)$ denote the analytic solution $u = 1/\alpha(z)$ of the equation $G(z,u) = 1$ for z in a sufficiently small (complex) neighbourhood of $z_0 = 1$ such that $\alpha(1) = \alpha$, then,

$$E[X_N] = \mu\frac{\log N}{\log\alpha} + O(1)$$

and

$$V[X_N] = \sigma^2\frac{\log N}{\log\alpha} + O(1),$$

where

$$\mu = \frac{\alpha'(1)}{\alpha} \quad \text{and} \quad \sigma^2 = \frac{\alpha''(1)}{\alpha} + \mu - \mu^2.$$

Proposition 3. *Let $G = (G_j)_{j\ge0}$ be as in (6). If $\sigma^2 \ne 0$, then, given an arbitrary small $\varepsilon > 0$, uniformly for all integers $k \ge 0$,*

$$\#\{n \le N : s_G(n) = k\} = \frac{N}{\sqrt{2\pi V[X_N]}}\left(\exp\left\{-\frac{(k - E[X_N])^2}{2V[X_N]}\right\} + O\left(\frac{1}{\log^{\frac12-\varepsilon} N}\right)\right).$$

Proof. This is Theorem 2.2 in the paper of Drmota and Gajdosik [5]. □

Let a be a positive integer. Let $q = -a + i$ (or $q = -a - i$) and set $Q = a^2 + 1$ and $\mathcal{N} = \{0, 1, \ldots, Q-1\}$. It is well known that every Gaussian integer z can be written uniquely as

$$z = \sum_{\ell \geq 0} \epsilon_\ell(z) q^\ell \qquad \text{with each } \epsilon_\ell \in \mathcal{N}.$$

Then, define the sum of digits function $s_q(z)$ of $z \in \mathbb{Z}[i]$ in base q as

$$s_q(z) = \sum_{\ell \geq 0} \epsilon_\ell(z).$$

Proposition 4. *Let $\mathcal{A}$ be the set of those positive integers a for which if $p \mid q = -a \pm i$ and $|p| \neq 1$, then $|p|^2 \geq 689$. Let $\mathcal{D}_N = \{z \in \mathbb{C} : |z| \leq \sqrt{N}\} \cap \mathbb{Z}[i]$ or $\mathcal{D}_N = \{z \in \mathbb{C} : |\Re(z)| \leq \sqrt{N}, |\Im(z)| \leq \sqrt{N}\} \cap \mathbb{Z}[i]$. Then, uniformly for all integers $k \geq 0$, we have*

$$\frac{1}{\#\mathcal{D}_N} \#\{z \in \mathcal{D}_N : s_q(z^2) = k\} = \frac{Q(k, q-1)}{\sqrt{2\pi\sigma_Q^2 \log_Q(N^2)}} \left(\exp\{-\frac{\Delta_k^2}{2}\} + O\left(\frac{(\log\log N)^{11}}{\sqrt{\log N}} \right) \right),$$

where

$$\Delta_k = \frac{k - \mu_Q \log_Q(N^2)}{\sqrt{\sigma_Q^2 \log_Q(N^2)}}, \quad \mu_Q = \frac{Q-1}{2}, \quad \sigma_Q^2 = \frac{Q^2-1}{12}.$$

Proof. This result is a simplified version of Theorem 4 in Morgenbesser [15]. □

Let $a \in \mathbb{N}$ and $q = -a + i \in \mathbb{Z}[i]$. Set $\mathcal{N} = \{0, 1, \ldots, a^2\}$. Then, every $z \in \mathbb{Z}[i]$ can be written uniquely as

$$z = \sum_{j \geq 0} \epsilon_j(z) q^j \qquad \text{with each } \epsilon_j(z) \in \mathcal{N}.$$

Let L be a non negative integer and consider a function $F : \mathcal{N}^{L+1} \to \mathbb{Z}$ satisfying $F(0, 0, \ldots, 0) = 0$ and set

$$s_F(z) = \sum_{j=-L}^{\infty} F(\epsilon_j(z), \epsilon_{j+1}(z), \ldots, \epsilon_{j+L}(z)).$$

The following is due to Drmota, Grabner and Liardet [6].

Proposition 5. *Under certain conditions on F stated in Corollary 3 in Drmota, Grabner and Liardet [6],*

$$\#\{z \in \mathbb{Z}[i] : |z|^2 < N,\ s_F(z) = k\} = \frac{\pi N}{\sqrt{2\pi\sigma^2 \log_{|q|^2} N}} \exp\left\{-\frac{(k - \mu \log_{|q|^2} N)^2}{2\sigma^2 \log_{|q|^2} N}\right\}\left(1 + O\left(\frac{1}{\sqrt{\log N}}\right)\right)$$

uniformly for $|k - \mu \log_{|q|^2} N| \le c\sqrt{\log_{|q|^2} N}$, where c can be taken arbitrarily large.

For any particular set of primes $\mathcal{P}$, let $E(x) = E_{\mathcal{P}}(x) := \sum_{\substack{p \le x \\ p \in \mathcal{P}}} \frac{1}{p}$.

The following two results, which we state as propositions, are due respectively to Halász [11] and Kátai [12].

Proposition 6. (Halász) *Let $0 < \delta \le 1$ and let $\mathcal{P}$ be a set of primes with corresponding functions $\Omega_{\mathcal{P}}(n)$ and $E(x) = E_{\mathcal{P}}(x)$. Then, assuming that $E(x) \to \infty$ as $x \to \infty$, the estimate*

$$\sum_{\substack{n \le x \\ \Omega_{\mathcal{P}}(n) = k}} 1 = \frac{xE(x)^k}{k!} e^{-E(x)} \left\{1 + O\left(\frac{|k - E(x)|}{E(x)}\right) + O\left(\frac{1}{\sqrt{E(x)}}\right)\right\}$$

holds uniformly for all positive integers k and real numbers $x \ge 3$ satisfying

$$E(x) \ge \frac{8}{\delta^3} \qquad \text{and} \qquad \delta \le \frac{k}{E(x)} \le 2 - \delta.$$

Proposition 7. (Kátai) *For $1 \le h \le x$, let*

$$A_k(x,h) := \sum_{\substack{x \le n \le x+h \\ \omega(n) = k}} 1, \qquad B_k(x) := \sum_{\substack{n \le x \\ \omega(n) = k}} 1,$$

$$\delta_k(x,h) := \frac{A_k(x,h)}{h} - \frac{B_k(x)}{x}, \qquad E(x,h) := \sum_{k=1}^{\infty} \delta_k^2(x,h).$$

Letting $\varepsilon > 0$ be an arbitrarily small number and $x^{7/12+\varepsilon} \le h \le x$, then

$$E(x,h) \ll \frac{1}{\log^2 x \cdot \sqrt{\log\log x}}.$$

3 Main results

Theorem 1. *Let $f(x)$ be as in (1), $h(n)$ be one of the five functions listed in (2) and $y_n := f(h(n))$. Then, the sequence $(y_n)_{n \ge 1}$ is uniformly distributed modulo 1.*

Theorem 2. *Let $f(x)$ be as in (1). Then, the sequence $(z_p)_{p\in\wp}$, where $z_p := f(s_q(p))$, is uniformly distributed modulo 1.*

Theorem 3. *Let $Q \geq 2$ and $q \geq 2$ be fixed integers. Let α be a strongly Q-normal number. Let g be a real valued continuous function defined on $[0,1]$ such that $\int_0^1 g(x)\,dx = 0$. Then,*

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} g(\alpha Q^{h(n)}) = 0, \tag{7}$$

where $h(n) = s_q(n)$ or $s_q(n^2)$. Moreover, letting $\pi(N)$ stand for the number of prime numbers not exceeding N, we have

$$\lim_{N\to\infty} \frac{1}{\pi(N)} \sum_{p\leq N} g(\alpha Q^{s_q(p)}) = 0. \tag{8}$$

The following corollary follows from estimate (7) of Theorem 3.

Corollary 1. *With α and $h(n)$ as in Theorem 3, the sequence $(\alpha Q^{h(p)})_{p\in\wp}$ is uniformly distributed modulo 1.*

In light of Proposition 3, we have the following two corollaries.

Corollary 2. *Let G be as in (4). Then, letting f be as in (1), the sequence $(\{f(s_G(n))\})_{n\geq 0}$ is uniformly distributed modulo 1.*

Corollary 3. *Let G be as in (4). Then, if α is a strongly normal number in base Q, the sequence $(\{\alpha \cdot Q^{s_G(n)}\})_{n\geq 0}$ is uniformly distributed modulo 1.*

As a direct consequence of the Main Lemma and of Proposition 4, we have the following result.

Theorem 4. *Let $\mathcal{D}_N$ be as in Proposition 4. Let f be as in (1). For each $z \in \mathcal{D}_N$, set $y_z := f(s_q(z^2))$. Then, the discrepancy of the sequence y_z tends to 0 as $N \to \infty$, that is*

$$D(y_z : z \in \mathcal{D}_N) \to 0 \qquad \text{as } N \to \infty.$$

Theorem 5. *Let $\mathcal{D}_N$ be as in Proposition 4. Let α be a strongly normal number in base Q and consider the sequence $(y_z)_{z\in\mathcal{D}_N}$. Then*

$$D(y_z : z \in \mathcal{D}_N) \to 0 \qquad \text{as } N \to \infty.$$

In line with Proposition 7, we have the following.

Theorem 6. *Let $\varepsilon > 0$ be a fixed number. Let $H = \lfloor x^{7/12+\varepsilon} \rfloor$ and set*

$$\pi_k([x, x+H]) := \#\{n \in [x, x+H] : \omega(n) = k\}.$$

Let f be as in (1) and set

$$S(x) = \sum_{x\leq n\leq x+H} e(f(\omega(n))).$$

Then

$$\frac{S(x)}{H} \to 0 \qquad \text{as } x \to \infty.$$

4 Preliminary lemmas

Lemma 1. *Let α be a non Liouville number and let $f(x)$ be as in (1). Then,*

$$\sup_{U\ge 1}\frac{1}{N}\left|\sum_{n=U+1}^{U+N} e(f(n))\right| \to 0 \qquad \text{as } N\to\infty.$$

Proof. Since α is a non Liouville number, there exists a positive integer ℓ such that if τ is a fixed positive number and

$$\left|\alpha-\frac{t}{s}\right| \le \frac{1}{s\tau}, \quad (t,s)=1, \quad s\le\tau,$$

then $\tau^{1/\ell} < s$.

Vaughan ([16], Lemma 2.4) proved that if $\left|\alpha-\frac{t}{s}\right| < \frac{1}{s^2}$ and $K=2^{t-1}$, then, given any small number $\varepsilon>0$,

$$\sum_{n=U+1}^{U+N} e(f(n)) \ll_\varepsilon N^{1+\varepsilon}\left(\frac{1}{s}+\frac{1}{N}+\frac{s}{N^t}\right)^{1/K}. \tag{9}$$

Now, choose $\tau = N^{t/2}$ so that $N^{t/2^\ell} < s < \tau$. It then follows from (9) that

$$\sum_{n=U+1}^{U+N} e(f(n)) \ll N^{1-\delta},$$

for some $\delta>0$ which depends only on ε and ℓ, thus completing the proof of Lemma 1. □

Using this result, we can establish our Main Lemma.

Lemma 2. *(**Main Lemma**) For each positive integer N, let $(E_N(k))_{k\ge1}$ be a sequence of non negative integers called weights which, given any $\delta>0$, satisfies the following three conditions:*

(a) $\displaystyle\sum_{k=1}^{\infty} E_N(k) = 1;$

(b) there exists a sequence $(L_N)_{N\ge1}$ which tends to infinity as $N\to\infty$ such that

$$\limsup_{N\to\infty} \sum_{\substack{k=1\\ \frac{|k-L_N|}{\sqrt{L_N}}>\frac{1}{\delta}}}^{\infty} E_N(k) \to 0 \qquad \text{as } \delta\to 0;$$

(c) $\displaystyle\lim_{N\to\infty} \max_{\frac{|k-L_N|}{\sqrt{L_N}}\le\frac{1}{\delta}} \max_{1\le\ell\le\delta^{3/2}} \left|\frac{E_N(k+\ell)}{E_N(k)}-1\right| = 0.$

Moreover, let α and f be as in (1) and let

$$T_N(f) := \sum_{k=1}^{\infty} e(f(k))E_N(k).$$

Then,

$$T_N(f) \to 0 \qquad \text{as } N \to \infty. \tag{10}$$

Proof. Let $\delta > 0$ be fixed and set

$$S := \lfloor \delta^{3/2}\sqrt{L_N} \rfloor, \quad t_m = \lfloor L_N \rfloor + mS \ \ (m = 1, 2, \ldots),$$
$$U_m = [t_m, t_{m+1} - 1] \ \ (m = 1, 2, \ldots).$$

Let us now write

$$T_N(f) = S_1(N) + S_2(N), \tag{11}$$

where

$$\begin{aligned} S_2(N) &= \sum_{|k - L_N| > \frac{1}{\delta}\sqrt{L_N}} E_k(N)e(f(k)), \\ S_1(N) &= \sum_{|m| \le 1/\delta^{5/2}} \sum_{k \in U_m} E_k(N)e(f(k)) = \sum_{|m| \le 1/\delta^{5/2}} S_1^{(m)}(N), \end{aligned}$$

say.

First observe that, by condition (b) above,

$$|S_2(N)| \le \sum_{\frac{|k - L_N|}{\sqrt{L_N}} > \frac{1}{\delta}} E_N(k) = o(1) \qquad \text{as } N \to \infty. \tag{12}$$

On the other hand, it follows from condition (c) above and Lemma 1 that, as $N \to \infty$,

$$\begin{aligned} \left| S_1^{(m)}(N) \right| &\le E_{t_m}(N) \left| \sum_{k \in U_m} e(f(k)) \right| + o(1) \sum_{k \in U_m} E_k(N) \\ &= o(1) S E_{t_m}(N) + o(1) \sum_{k \in U_m} E_k(N), \end{aligned}$$

while

$$\left| S E_{t_m}(N) - \sum_{k \in U_m} E_k(N) \right| = o(1) \sum_{k \in U_m} E_k(N).$$

Gathering these two estimates, we obtain that

$$S_1(N) \to 0 \quad \text{as } N \to \infty. \tag{13}$$

Using (12) and (13) in (11), conclusion (10) follows. □

Lemma 3. *For each integer $k \geq 1$, let*

$$\begin{aligned} \pi_k(x) &:= \#\{n \leq x : \omega(n) = k\}, \\ \pi_k^*(x) &:= \#\{n \leq x : \Omega(n) = k\} \end{aligned}$$

Then, the relations

$$\begin{aligned} \pi_k(x) &= (1+o(1))\frac{x}{\log x}\frac{(\log\log x)^{k-1}}{(k-1)!}, \\ \pi_k^*(x) &= (1+o(1))\frac{x}{\log x}\frac{(\log\log x)^{k-1}}{(k-1)!} \end{aligned}$$

hold uniformly for

$$|k - \log\log x| \leq \frac{1}{\delta_x}\sqrt{\log\log x}, \tag{14}$$

where δ_x is some function of x chosen appropriately and which tends to 0 as $x \to \infty$.

Proof. This follows from Theorem 10.4 stated in the book of De Koninck and Luca [3]. □

5 Proof of Theorem 1

We first consider the case when $h(n)$ is one of the three functions $\omega(n)$, $\Omega(n)$ and $\Omega_E(n)$. Set

$$\begin{aligned} \pi_k(N) &= \#\{n \leq N : \omega(n) = k\}, \\ \pi_k^*(N) &= \#\{n \leq N : \Omega(n) = k\}, \\ T_k(N) &= \#\{n \leq N : \Omega_E(n) = k\}. \end{aligned}$$

In light of Lemma 3 and Proposition 6, the corresponding weights of the sequences $(\pi_k(N))_{k\geq 1}$, $(\pi_k^*(N))_{k\geq 1}$ and $(T_k(N))_{k\geq 1}$ are $\pi_k(N)/N$, $\pi_k^*(N)/N$ and $T_k(N)/N$, respectively.

Now, in order to obtain the conclusion of the Theorem, we only need to prove that, for each non zero integer m,

$$\frac{1}{N}\sum_{n\leq N} e(mf(h(n))) \to 0 \qquad \text{as } N \to \infty.$$

But this is guaranteed by Lemma 1 if we take into account the fact that since α is a non Liouville number, the number $m\alpha$ is also non Liouville for each $m \in \mathbb{Z}\setminus\{0\}$. Hence, the theorem is proved.

6 Proof of Theorem 2

We cannot make a direct use of Lemma 2 because the estimate in that lemma only holds for those positive integers k such that $(k, q-1) = 1$. To avoid this obstacle, we shall subdivide the positive integers k according to their residue class modulo $q-1$. Observe that there are $\varphi(q-1)$ such classes. Hence, we write each k as

$$k = t(q-1) + \ell, \qquad (\ell, q-1) = 1.$$

Hence, for each positive integer ℓ such that $(\ell, q-1) = 1$, we set

$$\wp_\ell := \{p \in \wp : s_q(p) \equiv \ell \pmod{q-1}\}, \qquad \Pi_\ell(N) := \#\{p \le N : p \in \wp_\ell\}. \quad (15)$$

It is easy to verify that

$$\frac{\Pi_\ell(N)}{\pi(N)} = (1 + o(1))\frac{1}{\varphi(q-1)} \qquad (N \to \infty). \quad (16)$$

Thus, in order to prove Theorem 2, we need to show that the sum

$$U_\ell(N) := \sum_{\substack{p \le N \\ s_q(p) \equiv \ell \pmod{q-1}}} e(mf(s_q(p))),$$

where m is any fixed non zero integer, satisfies

$$U_\ell(N) = o(1) \quad \text{as } N \to \infty. \quad (17)$$

Setting

$$\sigma_N(k) := \#\{p \le N : s_q(p) = k\},$$

we have

$$\begin{aligned} U_\ell(N) &= \sum_{k \equiv \ell \pmod{q-1}} e(mf(k))\sigma_N(k) \\ &= \sum_{t \ge 0} e(mf(t(q-1)+\ell))\sigma_N(t(q-1)+\ell). \end{aligned} \quad (18)$$

Observe that the leading coefficient of the above polynomial $f(t(q-1)+\ell)$ is $\alpha(q-1)^k$, which is a non Liouville number as well (as we mentioned in the proof of Theorem 1), and also that the functions

$$w_N(t) := \frac{1}{\Pi_\ell(N)}\sigma_N(t(q-1)+\ell)$$

may be considered as weights (since $\sum_{k=1}^{\infty} w_N(t) = 1$). Thus, applying Lemma 2, we obtain (17), thereby completing the proof of Theorem 2.

7 Proof of Theorem 3

We shall skip the proof of estimate (7), since it can be obtained along the same lines as that of the main theorem in De Koninck, Kátai and Phong [2].

In order to obtain (8), we separate the set $\wp$ into $\varphi(q-1)$ distinct sets $\wp_\ell$, with corresponding counting function $\Pi_N(\ell)$ defined in (15).

Observe that

$$g(\alpha Q^{t(q-1)+\ell})\sigma_N(t(q-1)+\ell) = g((\alpha Q^\ell)\cdot Q^{t(q-1)})\sigma_N(t(q-1)+\ell)$$

Now, since α is a strongly Q-normal number, then so is αQ^ℓ, a number which is strongly Q^{q-1}-normal.

We then have

$$\begin{aligned}\sum_{p\le N} g(\alpha Q^{s_q(p)}) &= \sum_{k\ge 1}\sum_{\substack{p\le N\\ s_q(p)=k}} g(\alpha Q^k)\\ &= \sum_{\substack{\ell=1\\(\ell,q-1)=1}}^{q-1}\sum_{\substack{p\le N\\ p\in\wp_\ell}} g(\alpha Q^{t(q-1)+\ell})\sigma_N(t(q-1)+\ell)\\ &= \sum_{\substack{\ell=1\\(\ell,q-1)=1}}^{q-1}\sum_{\substack{p\le N\\ p\in\wp_\ell}} g((\alpha Q^\ell)\cdot Q^{t(q-1)})\sigma_N(t(q-1)+\ell).\end{aligned}$$

Since we then have

$$\lim_{N\to\infty}\frac{1}{\Pi_\ell(N)}\sum_{\substack{p\le N\\ p\in\wp_\ell}} g(\alpha Q^{s_q(p)}) = 0 \quad \text{for each } \ell \text{ with } (\ell, q-1)=1,$$

summing up over all ℓ's such that $(\ell, q-1)=1$, estimate (8) follows immediately.

References

[1] J.M. De Koninck and I. Kátai, *On the distribution modulo 1 of the values of* $F(n)+\alpha\sigma(n)$, Publicationes Mathematicae Debrecen **66** (2005), 121–128.

[2] J.M. De Koninck, I. Kátai and B.M. Phong, *On strong normality*, preprint.

[3] J.M. De Koninck and F. Luca, *Analytic Number Theory: Exploring the Anatomy of Integers*, Graduate Studies in Mathematics, Vol. 134, American Mathematical Society, Providence, Rhode Island, 2012.

[4] H. Delange, *Sur la fonction sommatoire de la fonction "somme des chiffres"*, Enseign. Math. (2) **21**.1 (1975), 31–47.

[5] M. Drmota and J. Gajdosik, *The distribution of the sum-of-digits function*, J. Théor. Nombres Bordeaux **10** (1998), No. 1, 17–32.

[6] M. Drmota, P.J. Grabner and P. Liardet, *Block additive functions on the Gaussian integers*, Acta Arith. **135** (2008), No. 4, 299–332.

[7] M. Drmota and J. Rivat, *The sum of digits function of squares*, J. London Math. Soc. **72** (2005), 273–292.

[8] M. Drmota, C. Mauduit and J. Rivat, *Primes with an average sum of digits*, Compos. Math. **145** (2009), No. 2, 271–292.

[9] M. Drmota, C. Mauduit and J. Rivat, *The sum of digits function of polynomial sequences*, J. Lond. Math. Soc. (2) **84** (2011), No. 1, 81–102.

[10] A. O. Gelfond, *Sur des nombres qui ont des propriétés additives et multiplicatives données*, Acta Arith. **13** (1968), 259–265.

[11] G. Halász, *On the distribution of additive and the mean values of multiplicative arithmetic functions*, Studia Sci. Math. Hungar. **6** (1971), 211–233.

[12] I. Kátai, *A remark on a paper of K. Ramachandra* in Number theory (Ootacamund, 1984), 147–152, Lecture Notes in Math., 1122, Springer, Berlin, 1985.

[13] C. Mauduit and J. Rivat, *Sur un problème de Gelfond: la somme des chiffres des nombres premiers*, Ann. of Math. (2) **171** (2010), no. 3, 1591-1646.

[14] C. Mauduit and J. Rivat, *La somme des chiffres des carrés*, Acta Math. **203** (2009), No. 1, 107–148.

[15] J. F. Morgenbesser, *The sum of digits of squares in* $\mathbb{Z}[i]$, J. Number Theory **130** (2010), 1433–1469.

[16] R. C. Vaughan, *The Hardy-Littlewood method*, Bull. Amer. Math. Soc. (N.S.) **7** (1982), no. 2, 433–437.

Received 25th February 2015

Acta Cybernetica **22** (2015) 349–357.

Asymptotic Approximation for the Quotient Complexities of Atoms

Volker Diekert* and Tobias Walter*†

For Ferenc Gécseg, in memoriam

Abstract

In a series of papers, Brzozowski together with Tamm, Davies, and Szykuła studied the quotient complexities of atoms of regular languages [6, 7, 3, 4]. The authors obtained precise bounds in terms of binomial sums for the most complex situations in the following five cases: ($\mathcal{G}$): general, ($\mathcal{R}$): right ideals, ($\mathcal{L}$): left ideals, ($\mathcal{T}$): two-sided ideals and ($\mathcal{S}$): suffix-free languages. In each case let $\kappa_{\mathcal{C}}(n)$ be the maximal complexity of an atom of a regular language L, where L has complexity $n \geq 2$ and belongs to the class $\mathcal{C} \in \{\mathcal{G}, \mathcal{R}, \mathcal{L}, \mathcal{T}, \mathcal{S}\}$. It is known that $\kappa_{\mathcal{T}}(n) \leq \kappa_{\mathcal{L}}(n) = \kappa_{\mathcal{R}}(n) \leq \kappa_{\mathcal{G}}(n) < 3^n$ and $\kappa_{\mathcal{S}}(n) = \kappa_{\mathcal{L}}(n-1)$. We show that the ratio $\frac{\kappa_{\mathcal{C}}(n)}{\kappa_{\mathcal{C}}(n-1)}$ tends exponentially fast to 3 in all five cases but it remains different from 3. This behaviour was suggested by experimental results of Brzozowski and Tamm; and the result for $\mathcal{G}$ was shown independently by Luke Schaeffer and the first author soon after the paper of Brzozowski and Tamm appeared in 2012. However, proofs for the asymptotic behavior of $\frac{\kappa_{\mathcal{G}}(n)}{\kappa_{\mathcal{G}}(n-1)}$ were never published; and the results here are valid for all five classes above. Moreover, there is an interesting oscillation for all $\mathcal{C}$: for almost all n we have $\frac{\kappa_{\mathcal{C}}(n)}{\kappa_{\mathcal{C}}(n-1)} > 3$ if and only if $\frac{\kappa_{\mathcal{C}}(n+1)}{\kappa_{\mathcal{C}}(n)} < 3$.

1 Introduction and Preliminaries

Let Σ denote a finite non-empty alphabet, Σ^* the set of words over Σ and $1 \in \Sigma^*$ the empty word. A *language* L is a subset of Σ^*. A class of languages is called a *Boolean algebra* if it is closed under finite unions and complementation. By $L \subseteq \Sigma^*$ we denote a regular language with $\emptyset \neq L \neq \Sigma^*$. The set of regular languages is denoted by $\mathcal{G}$, because it is the "general" case, here. The set $\overline{L} = \Sigma^* \setminus L$ is the complement of L. The language L is a left, right or two-sided ideal if $L = \Sigma^* L$, $L = L\Sigma^*$ or $L = \Sigma^* L \Sigma^*$. A language L is suffix-free if $w \in L$ and $xw \in L$ implies $x = 1$. We denote by $\mathcal{L}, \mathcal{R}, \mathcal{T}$ and

*FMI, Universität Stuttgart, Universitätsstraße 38, 70569 Stuttgart, Germany.
E-mail: {diekert,walter}@fmi.uni-stuttgart.de

†The second author was supported by the German Research Foundation (DFG) under grant DI 435/6-1.

$\mathcal{S}$ the classes of $\mathcal{L}$eft ideals, $\mathcal{R}$ight ideals, $\mathcal{T}$wo-sided ideals and $\mathcal{S}$uffix-free languages, respectively.

For $x \in \Sigma^*$ denote by $L(x) = \{y \in \Sigma^* \mid xy \in L\}$ the (left) quotient of L by x. Frequently, a left quotient $L(x)$ is also denoted by $x^{-1}L$. We prefer the notation $L(x)$ because Σ^* acts naturally on the right; and then the formula for the action becomes $L(x) \cdot y = L(xy)$. Indeed, the classical Myhill-Nerode Theorem asserts that this action leads to the minimal deterministic finite automaton accepting L. The set of states for this DFA is $Q_L = \{L(x) \mid x \in \Sigma^*\}$, the initial state is $L = L(1)$ and the final states are those $L(x)$ with $1 \in L(x)$. The transitions are given by $L(x) \cdot a = L(xa)$ for $x \in \Sigma^*$ and $a \in \Sigma$. The size $|Q_L|$ is therefore the number of quotients of L. It is also called the *quotient complexity*, or simply the *complexity*, of L; and the complexity of L is denoted by $\kappa(L)$.

Given a regular language L it is natural to consider the smallest Boolean algebra $\mathrm{BQ}(L)$ which contains L and is closed under quotients. A priori, it is not obvious that $\mathrm{BQ}(L)$ is finite; but it is: every set in $\mathrm{BQ}(L)$ can be written as a union of *atoms* A_S where $S \subseteq Q_L$ and

$$A_S = \bigcap_{L(x) \in S} L(x) \cap \bigcap_{L(y) \notin S} \overline{L(y)}.$$

Atoms have been introduced by Brzozowski and Tamm in [5, 2]. The complexity of atoms was studied in [6, 7].

More generally, for $X, Y \subseteq Q_L$ define

$$L(X,Y) = \bigcap_{L(x) \in X} L(x) \cap \bigcap_{L(y) \in Y} \overline{L(y)}.$$

In particular, $A_S = L(S, Q_L \setminus S)$.

The observation $L(X,Y)(w) = L(X(w), Y(w)) = L(X', Y')$ with $X' = \{L(xw) \mid L(x) \in X\}$ and $Y' = \{L(xw) \mid L(x) \in Y\}$ leads to the following remark.

Remark 1.1. Let L be regular, n its complexity and $X, Y \subseteq Q_L$. Then the following assertions hold.

- $X \cap Y \neq \emptyset$ implies $L(X,Y) = \emptyset$.
- The non-empty quotients of A_S have the form $L(X,Y)$ with $|X| \leq |S|$ and $X \cap Y = \emptyset$.
- $S \neq T$ implies $A_S \cap A_T = \emptyset$.
- Since $|\{A_S \mid S \subseteq Q_L\}| \leq 2^n$ and since every element in $\mathrm{BQ}(L)$ is a union of atoms, we have $|\mathrm{BQ}(L)| \leq 2^{2^n}$. The upper bound 2^{2^n} is optimal: It is proved in [6] that for every $n \geq 2$ there exists a language L of complexity n with 2^n atoms. As $A_S \cap A_T = \emptyset$ for $S \neq T$, the atoms form a partition of Σ^*. Hence, there are 2^{2^n} distinct unions of atoms.

A 3-*coloring* of Q is a disjoint union $Q = X \cup Y \cup W$ where X, Y, W are called colors. Thus, there are 3^n different 3-colorings. A combinatorial interpretation leads to the well-known formula

$$3^n = \sum_{x=0}^{n} \sum_{y=0}^{n-x} \binom{n}{x} \binom{n-x}{y}.$$

Indeed, each 3-coloring is uniquely described by first choosing the elements with color X out of n elements and then choosing the elements with color Y out of the remaining $n - |X|$ elements. As $X \cap Y = \emptyset$ induces a unique 3-coloring with $W = Q \setminus (X \cup Y)$, there are at most 3^n non-empty sets of the form $L(X, Y)$. We will use the concept of 3-colorings in order to give a combinatorial interpretation for the bounds of [3].

2 Upper bounds

In this section we will deduce simple upper bounds for the complexity of atoms in each case by making observations on the structure of the quotients. These upper bounds are not optimal, but straightforward and still good enough to show the asymptotic behaviour.

Lemma 2.1. *Let L be a regular language of complexity $n \geq 2$ and A_S be an atom of L. Then A_S has complexity of at most $3^n + 1$.*

Proof. There are at most 3^n quotients of the form $L(X, Y)$ and the empty set. □

Lemma 2.2. *Let L be a right ideal of complexity $n \geq 2$ and A_S be an atom of L. Then A_S has complexity of at most 3^{n-1}.*

Proof. For all x with $1 \in L(x)$ we have $1 \cdot w \in L\Sigma^*(x) = L(x)$ for all $w \in \Sigma^*$ and, thus, $L(x) = \Sigma^*$. Therefore, Σ^* is the unique final state in Q_L. Additionally, we must have $\Sigma^* \in S$, as $\Sigma^* \notin S$ implies $A_S = \emptyset$. By $\Sigma^*(x) = \Sigma^*$ for all $x \in \Sigma^*$, we see that every quotient $A_S(x) = L(X, Y)$ must contain Σ^* in X. Thus, there are at most 3^{n-1} quotients $A_S(x)$, which shows that A_S has complexity of at most 3^{n-1}. □

Lemma 2.3. *Let L be a left ideal of complexity $n \geq 2$ and A_S be an atom of L. Then A_S has complexity of at most $3^{n-1} + 2$.*

Proof. As $L = \Sigma^* L$, we have

$$L \subseteq L(x) = \{y \in \Sigma^* \mid xy \in L\} = \{y \in \Sigma^* \mid xy \in \Sigma^* L\}$$

for all $x \in \Sigma^*$. Hence, for any X with $L \in X$ we have

$$L(X, Y) = L \cap \bigcap_{L(y) \in Y} \overline{L(y)}.$$

Thus, if $Y \neq \emptyset$ then $L(X, Y) = \emptyset$. Also, $L \subseteq L(x)$ implies $\overline{L(x)} \subseteq \overline{L}$ which yields $L(X, Y) = L(X, Y \cup \{L\})$ for $L \notin X$. It follows that there are at most $3^{n-1} + 2$ quotients.

The first term counts the $L(X, Y)$ with $X = \{L\}$ which is not smaller than to count the $L(X, Y)$ with $L \in X$. By the argument above, only $L(\{L\}, \emptyset)$ and $\emptyset$ are of this type.

The second term counts those (X, Y, W) with $L \notin X$ (in which case we can assume $L \in Y$ by the argumentation above).

□

Lemma 2.4. *Let L be a two-sided ideal of complexity $n \geq 2$ and A_S be an atom of L. Then A_S has complexity of at most $3^{n-2}+2$.*

Proof. This is similar to the analysis in the case of left ideals, since there are only two cases with $L \in X$. Again, for $L \notin X$ we have $L(X,Y) = L(X, Y \cup \{L\}))$, i.e., we may assume $L \in Y$. As every two-sided ideal is in particular a right ideal, we have that Σ^* is the unique final state in Q_L. Again, only those $L(X,Y)$ with $\Sigma^* \in X$ are reachable as quotients of an atom. Thus, we can deduce that A_S has at most $3^{n-2}+2$ quotients. □

3 Lower bounds

In this section we revisit the complexity bounds of atoms for left, right and two-sided ideals obtained by Brzozowski, Tamm and Davies. The bounds are optimal. We use them to derive (weaker) lower bounds in explicit form. For $|S| \in \mathcal{O}(1)$ or $n - |S| \in \mathcal{O}(1)$ it holds $\kappa(A_S) \in \mathcal{O}(2^n)$ where A_S is an atom of a language L of complexity n. As we are only interested in the maximal complexity of atoms of some language L, we will restrict the proposition below to $0 < |S| < n-1$. This excludes special cases not needed in our analysis.

Proposition 3.1 ([7, 1, 3])**.** *Let $k, n \in \mathbb{N}$ with $0 < k < n-1$ and $\mathcal{C} \in \{\mathcal{G}, \mathcal{R}, \mathcal{L}, \mathcal{T}\}$. Then there there exists a language $L \in \mathcal{C}$ of complexity n and an atom A_S of L with $|S| = k$ such that the complexity of A_S is given by:*

$$\kappa(A_S) = \begin{cases} 1 + \sum_{x=1}^{|S|} \sum_{y=1}^{n-|S|} \binom{n}{x}\binom{n-x}{y}, & \text{for } \mathcal{C} = \mathcal{G} \\ 1 + \sum_{x=1}^{|S|} \sum_{y=1}^{n-|S|} \binom{n-1}{x-1}\binom{n-x}{y}, & \text{for } \mathcal{C} = \mathcal{R} \\ 1 + \sum_{x=1}^{|S|} \sum_{y=1}^{n-|S|} \binom{n-1}{x}\binom{n-x-1}{y-1}, & \text{for } \mathcal{C} = \mathcal{L} \\ 1 + \sum_{x=1}^{|S|} \sum_{y=1}^{n-|S|} \binom{n-2}{x-1}\binom{n-x-1}{y-1}, & \text{for } \mathcal{C} = \mathcal{T}. \end{cases}$$

Moreover, for every L of complexity n in the corresponding class $\mathcal{C}$ and every S, the right hand sides are upper bounds.

Remark 3.1. The maximal complexity of atoms of left ideals and right ideals turns out to be same. This was also observed in [3]. Indeed, using the trinomial revision (see for example [8]) for the last equality below, we can do the following calculation:

$$\begin{aligned} \sum_{x=1}^{|S|} \sum_{y=1}^{n-|S|} \binom{n-1}{x-1}\binom{n-x}{y} &= \sum_{y=1}^{n-|S|} \sum_{x=1}^{|S|} \binom{n-1}{x-1}\binom{n-x}{y} \\ &= \sum_{x=1}^{n-|S|} \sum_{y=1}^{|S|} \binom{n-1}{y-1}\binom{n-y}{x} \\ &= \sum_{x=1}^{n-|S|} \sum_{y=1}^{|S|} \binom{n-1}{x}\binom{n-x-1}{y-1}. \end{aligned}$$

In the following we give a combinatorial interpretation of the sums in Proposition 3.1.

Lemma 3.1. *For every $n \geq 3$ there exists a regular language L of complexity n such that L has an atom A_S of complexity in $3^n - \Theta(8^{n/2})$.*

Proof. Let S be such that $|S| = n/2$ (if n is even; the proof is similar if n is odd). By Proposition 3.1 there exists a regular language L of complexity n such that the atom A_S of L has complexity $1+\sum_{x=1}^{n/2}\sum_{y=1}^{n/2}\binom{n}{x}\binom{n-x}{y}$ for some $S \subseteq Q_L$. Observe that $\sum_{x=0}^{n}\sum_{y=0}^{n-x}\binom{n}{x}\binom{n-x}{y} = 3^n$ has the combinatorial interpretation of counting all 3-colorings of $Q = X \cup Y \cup W$. We will count the 3-colorings which are missing in $\sum_{x=1}^{n/2}\sum_{y=1}^{n/2}\binom{n}{x}\binom{n-x}{y}$. As the indices start with 1 instead of 0 and end with $n/2$ instead of n, the cases for $X = \emptyset$ or $Y = \emptyset$ and for $|X| > n/2$ or $|Y| > n/2$ are missing. There are 2^n possibilities with $|X| = 0$ and 2^n many with $|Y| = 0$. There are at most 2^n possibilities for X with $|X| > n/2$. Since $|X| > n/2$, we must have $|Y| < n/2$ and, thus, there are at most $2^{n/2}$ choices remaining for Y. This leaves at most $2^n \cdot 2^{n/2} = 8^{n/2}$ missing 3-colorings with $|X| > n/2$. The case $|Y| > n/2$ is symmetrical. Combining all those cases shows that the number of missing 3-colorings is in $\Theta(8^{n/2})$. □

Lemma 3.2. *For every $n \geq 3$ there exists a right ideal L of complexity n such that L has an atom A_S of L with complexity in $3^{n-1} - \Theta(8^{n/2})$.*

Proof. By Proposition 3.1 we obtain a right ideal L of complexity n such that L has an atom A_S of complexity $1 + \sum_{x=1}^{n/2}\sum_{y=1}^{n/2}\binom{n-1}{x-1}\binom{n-x}{y}$. Observe that $\sum_{x=1}^{n}\sum_{y=0}^{n-x}\binom{n-1}{x-1}\binom{n-x}{y} = 3^{n-1}$ has the combinatorial interpretation of counting 3-colorings of $Q = X \cup Y \cup W$ with a precolored element $\Sigma^* \in X$ (see Section 2 on why Σ^* is in X). Again, we count the 3-colorings which are missing in $\sum_{x=1}^{n/2}\sum_{y=1}^{n/2}\binom{n-1}{x-1}\binom{n-x}{y}$; namely, those with $Y = \emptyset$, $|X| > n/2$ or $|Y| > n/2$. The analysis in the proof of Lemma 3.1 shows that this is in $\Theta(8^{n/2})$. □

Lemma 3.3. *For every $n \geq 3$ there exists a two-sided ideal L of complexity n such that there is an atom A_S of L with complexity in $3^{n-2} - \Theta(8^{n/2})$.*

Proof. By Proposition 3.1 we obtain a two-sided ideal L of complexity n such that L has an atom A_S of complexity $1 + \sum_{x=1}^{n/2}\sum_{y=1}^{n/2}\binom{n-2}{x-1}\binom{n-x-1}{y-1}$. We count the number of 3-colorings of $Q = X \cup Y \cup W$ with precolored elements $L \in Y$ and $\Sigma^* \in X$. There are $3^{n-2} = \sum_{x=1}^{n}\sum_{y=1}^{n-x}\binom{n-2}{x-1}\binom{n-x-1}{y-1}$ such 3-colorings. Thus, in $\sum_{x=1}^{n/2}\sum_{y=1}^{n/2}\binom{n-2}{x-1}\binom{n-x-1}{y-1}$ the 3-colorings with $|X| > n/2$ or $|Y| > n/2$ are not counted. The analysis in the proof of Lemma 3.1 shows that this is in $\Theta(8^{n/2})$. □

4 Asymptotic behaviour

As above, let $\mathcal{C}$ be one of the classes: ($\mathcal{G}$) general regular languages, ($\mathcal{R}$) right ideals, ($\mathcal{L}$) left ideals, ($\mathcal{T}$) two-sided ideals or ($\mathcal{S}$) suffix-free languages. Define

$$\kappa_{\mathcal{C}}(n) = \max\{\kappa(A_S) \mid A_S \text{ is an atom of } L \in \mathcal{C} \text{ of complexity } n\}.$$

This section studies the behaviour of $\kappa_{\mathcal{C}}(n)/\kappa_{\mathcal{C}}(n-1)$ as a function in n.

n	8	9	10	11	12	13	14	15
$\kappa_{\mathcal{G}}(n)$	5083	15361	48733	146169	455797	1364091	4212001	12601332
ratio	3.284	3.022	3.173	2.999	3.118	2.992	3.088	2.992

Table 1: $\kappa_{\mathcal{G}}(n)$ and the ratio $\kappa_{\mathcal{G}}(n)/\kappa_{\mathcal{G}}(n-1)$ for some small n

4.1 Asymptotic Approximation

Combining the explicit lower and upper bounds we obtain the following result which was announced in [3].

Theorem 4.1. *Let $\mathcal{C} \in \{\mathcal{G}, \mathcal{L}, \mathcal{R}, \mathcal{T}, \mathcal{S}\}$. Then the ratio $\kappa_{\mathcal{C}}(n)/\kappa_{\mathcal{C}}(n-1)$ converges exponentially fast to 3.*

Proof. First, we will prove this for the class of right ideals. By Lemma 3.2 and Lemma 2.2 we have

$$3^{n-1} - f(n) \leq \kappa_{\mathcal{R}}(n) \leq 3^{n-1}$$

for some $f \in \Theta(8^{n/2})$. We conclude

$$\frac{3^{n-1} - f(n)}{3^{n-2}} \leq \frac{\kappa_{\mathcal{R}}(n)}{\kappa_{\mathcal{R}}(n-1)} \leq \frac{3^{n-1}}{3^{n-2} - f(n-1)},$$

which implies the assertion. The cases of general regular languages and two-sided ideals are analogous using the respective lemmas. The case of left ideals follows as $\kappa_{\mathcal{L}}(n) = \kappa_{\mathcal{R}}(n)$ for $n \geq 3$ by Remark 3.1. The case of suffix-free languages is clear because $\kappa_{\mathcal{S}}(n) = \kappa_{\mathcal{L}}(n-1)$ as is shown in [4]. □

4.2 Oscillation

In [6] it is shown that

$$\kappa_{\mathcal{G}}(n) = 1 + \sum_{x=1}^{\lfloor n/2 \rfloor} \sum_{y=1}^{n-\lfloor n/2 \rfloor} \binom{n}{x} \binom{n-x}{y}. \tag{1}$$

This means that $\kappa(A_S)$ is maximal for $|S| = \lfloor n/2 \rfloor$. In this section we will prove that the quotient $\kappa_{\mathcal{C}}(n)/\kappa_{\mathcal{C}}(n-1)$ does not only converge to 3, but also does so oscillating. Oscillation was observed first by calculating $\kappa_{\mathcal{G}}(n)$ in the range $1 \leq n \leq 20$. It came as a little surprise as the first ten values do not reveal this, [6]. In Table 1 we display the values $\kappa_{\mathcal{G}}(n)$ and the ratios $\kappa_{\mathcal{G}}(n)/\kappa_{\mathcal{G}}(n-1)$ for $8 \leq n \leq 15$.

Theorem 4.2. *For every $\mathcal{C} \in \{\mathcal{G}, \mathcal{R}, \mathcal{L}, \mathcal{T}, \mathcal{S}\}$ there exists some $n_0 \in \mathbb{N}$ such that*

$$\kappa_{\mathcal{C}}(n)/\kappa_{\mathcal{C}}(n-1) > 3 \iff \kappa_{\mathcal{C}}(n+1)/\kappa_{\mathcal{C}}(n) < 3$$

for all $n \geq n_0$. Moreover, for almost all n we have $\kappa_{\mathcal{C}}(n)/\kappa_{\mathcal{C}}(n-1) \neq 3$.

Proof. We give the proof for the general class $\mathcal{C} = \mathcal{G}$, only. Similar calculations show the result in the other cases. This is not done here and left to the reader.

We apply the interpretation of the sums as the number of 3-colorings from above. Let $\mathrm{HC_n}$ be the set of all 3-colorings of $\{1, \ldots, n\}$ in which

the color X appears at most $\lfloor n/2 \rfloor$ times and the color Y appears at most $n - \lfloor n/2 \rfloor = \lceil n/2 \rceil$ times. We also let $\mathrm{hc}(n) = |\mathrm{HC_n}|$.

Besides the term $+1$ and starting at $x = 1$ and $y = 1$, instead of $x = 0$ and $y = 0$ for $\mathrm{hc}(n)$, the right-hand side in Equation (1) is identical to $\mathrm{hc}(n)$. More precisely, we have the following estimation.

$$\kappa_{\mathcal{G}}(n) < \mathrm{hc}(n) = \sum_{x=0}^{\lfloor n/2 \rfloor} \sum_{y=0}^{n-\lfloor n/2 \rfloor} \binom{n}{x} \binom{n-x}{y} \leq \kappa_{\mathcal{G}}(n) + 2^{n+1}. \qquad (2)$$

Thus, apart from an error term bounded by $2^{n+1} \in \mathcal{O}(2^n)$ the numbers $\kappa_{\mathcal{G}}(n)$ and $\mathrm{hc}(n)$ are equal. We show two statements.

1. If n is large enough and even, then $\kappa_{\mathcal{G}}(n+1) < 3 \cdot \kappa_{\mathcal{G}}(n)$.
2. If n is large enough and odd, then $\kappa_{\mathcal{G}}(n+1) > 3 \cdot \kappa_{\mathcal{G}}(n)$.

1.) Let n be even, i.e., $n/2 = \lceil n/2 \rceil = \lfloor n/2 \rfloor = \lfloor (n+1)/2 \rfloor$ and $\lceil n/2 \rceil + 1 = \lceil (n+1)/2 \rceil$. We calculate $\mathrm{hc}(n+1)$ by considering 3-colorings of $\{1, \ldots, n\}$ and extending them by choosing a color for $n+1$. Consider first any 3-coloring of $\{1, \ldots, n\}$ in $\mathrm{HC_n}$. There are 3 possible extensions of this 3-coloring by choosing the color of $n+1$, i.e., there are at most $3\mathrm{hc}(n)$ possible extensions of $\mathrm{HC_n}$. Not all of those extensions are in $\mathrm{HC_{n+1}}$. We cannot extend those 3-colorings of $\{1, \ldots, n\}$, which already had $n/2$ elements in X by choosing $n+1 \in X$. Let us count how many such 3-colorings in $\mathrm{HC}_(n)$ exist: there are $\binom{n}{n/2}$ choices for X and, for each fixed X, there are $\sum_{y=0}^{n/2} \binom{n/2}{y} = 2^{n/2}$ choices for Y. In total, we see that there are $3\mathrm{hc}(n) - \binom{n}{n/2} 2^{n/2}$ extensions of $\mathrm{HC_n}$ in $\mathrm{HC_{n+1}}$.

It remains to count the number of 3-colorings in $\mathrm{HC_{n+1}}$ which are not extensions of any 3-coloring in $\mathrm{HC_n}$. These are exactly the extensions of those 3-colorings of $\{1, \ldots, n\}$ in which we have $|Y| = n/2 + 1$. As $n - (n/2 + 1) = n/2 - 1$, X may contain at most $n/2 - 1$ elements, i.e., $|X| \leq n/2 - 1$. Consequently, $n+1$ may be either colored X or W. Thus, there are $2 \cdot \binom{n}{n/2+1} 2^{n/2-1} = \binom{n}{n/2+1} 2^{n/2}$ extensions of this type. The binomial coefficient $\binom{n}{n/2}$ is the largest one among all $\binom{n}{k}$ where $k \in \mathbb{Z}$. In particular, $\binom{n}{n/2} \geq \frac{2^n}{n+1}$ for all $n \in \mathbb{N}$ and $\binom{n}{n/2} \geq \frac{2^n}{n}$ for $n \geq 2$. We conclude

$$\begin{aligned} 3\mathrm{hc}(n) - \mathrm{hc}(n+1) &= 2^{n/2} \left(\binom{n}{n/2} - \binom{n}{n/2+1} \right) \\ &= 2^{n/2} \binom{n}{n/2} \cdot \frac{1}{n/2+1} \\ &\geq 2^{n/2} \cdot 2^n \cdot \frac{1}{n \cdot (n/2+1)} = \frac{\sqrt{8}^n}{n \cdot (n/2+1)}. \end{aligned}$$

Note that the term $\binom{n}{n/2} - \binom{n}{n/2+1} = \binom{n}{n/2} \cdot \frac{1}{n/2+1}$ is equal to the Catalan number $C_{n/2}$; and better estimations for the difference $3\mathrm{hc}(n) - \mathrm{hc}(n+1)$ are possible. The fraction $\frac{\sqrt{8}^n}{n \cdot (n/2+1)}$ is greater than three times the error term 2^{n+1} for almost all n.

Class $\mathcal{C}$	n_0
regular languages ($\mathcal{G}$)	10
left ideals ($\mathcal{L}$)	11
right ideals ($\mathcal{R}$)	11
two-sided ideals ($\mathcal{T}$)	5
suffix-free languages ($\mathcal{S}$)	12

Table 2: Smallest n_0 where oscillation starts.

Thus, there exists a (small) number n_0 such that for all even $n \geq n_0$ we obtain $\kappa_{\mathcal{G}}(n+1) < 3\kappa_{\mathcal{G}}(n)$. According to Table 1 we have $n_0 = 10$.

2.) Let n be odd and $n \geq 3$. We have $(n+1)/2 = \lfloor n/2 \rfloor + 1 = \lceil n/2 \rceil$. Again, consider the extensions of 3-colorings of $\{1, \ldots, n\}$. First, consider the extensions of $\mathrm{HC_n}$. They are not in $\mathrm{HC_{n+1}}$ if and only if $|Y| = \lceil n/2 \rceil$ and the color of $n+1$ is the color Y. For fixed Y, there are $2^{\lfloor n/2 \rfloor}$ choices for X. In total, there are $3\mathrm{hc}(n) - \binom{n}{\lceil n/2 \rceil} 2^{\lfloor n/2 \rfloor}$ extensions of colorings in $\mathrm{HC_n}$ which are in $\mathrm{HC_{n+1}}$.

It remains to count the number of colorings in $\mathrm{HC_{n+1}}$ which are not extensions of colorings in $\mathrm{HC_n}$.

These are exactly the extensions of those 3-colorings of $\{1, \ldots, n\}$ in which we have $|X| = \lfloor n/2 \rfloor + 1$. As $n - (\lfloor n/2 \rfloor + 1) = \lceil n/2 \rceil - 1$, the color Y may contain at most $\lceil n/2 \rceil - 1$ elements, i.e., $|Y| \leq \lceil n/2 \rceil - 1$. Consequently, $n+1$ may be either colored Y or W. Thus, there are $2 \cdot \binom{n}{\lfloor n/2 \rfloor + 1} 2^{\lceil n/2 \rceil - 1} = 2 \cdot \binom{n}{\lfloor n/2 \rfloor + 1} 2^{\lfloor n/2 \rfloor}$ extensions of this type. Consequently, we obtain

$$\begin{aligned} \mathrm{hc}(n+1) - 3\mathrm{hc}(n) &= 2^{\lfloor n/2 \rfloor} \left(2 \binom{n}{\lfloor n/2 \rfloor + 1} - \binom{n}{\lceil n/2 \rceil} \right) \\ &= 2^{\lfloor n/2 \rfloor} \binom{n}{\lceil n/2 \rceil} \geq 2^{\lfloor n/2 \rfloor} 2^n / n. \end{aligned}$$

This number is asymptotically larger than any error in $\mathcal{O}(2^n)$ and, thus, we obtain $\kappa_{\mathcal{G}}(n+1) > 3\kappa_{\mathcal{G}}(n)$ for all odd n greater than some n_0. This concludes the proof of the oscillation property in the case of $\mathcal{C} = \mathcal{G}$. The other cases can be handled with very similar methods. Therefore, as mentioned above, this is left to the reader. □

We calculated the exact values for n_0 in every case, see Table 2. Note that in the cases ($\mathcal{G}$), ($\mathcal{L}$), ($\mathcal{R}$) and ($\mathcal{S}$) $\kappa(n)/\kappa(n-1) > 3$ holds for $4 \leq n < n_0$.

References

[1] J. Brzozowski and G. Davies. Most complex regular right-ideal languages. In H. Jürgensen, J. Karhumäki, and A. Okhotin, editors, *Descriptional Complexity of Formal Systems*, volume 8614 of *Lecture Notes in Computer Science*, pages 90–101. Springer International Publishing, 2014.

[2] J. Brzozowski and H. Tamm. Theory of átomata. *Theoretical Computer Science*, 539:13–27, 2014.

[3] J. A. Brzozowski and S. Davies. Quotient complexities of atoms in regular ideal languages. *Acta Cybernetica*, 22(2):293–311, 2015. Preliminary version: ArXiv e-prints, http://arxiv.org/abs/1503.02208.

[4] J. A. Brzozowski and M. Szykula. Complexity of suffix-free regular languages. In A. Kosowski and I. Walukiewicz, editors, *Fundamentals of Computation Theory - 20th International Symposium, FCT 2015, Gdańsk, Poland, August 17-19, 2015, Proceedings*, volume 9210 of *Lecture Notes in Computer Science*, pages 146–159. Springer, 2015.

[5] J. A. Brzozowski and H. Tamm. Theory of átomata. In G. Mauri and A. Leporati, editors, *Developments in Language Theory - 15th International Conference, DLT 2011, Milan, Italy, July 19-22, 2011. Proceedings*, volume 6795 of *Lecture Notes in Computer Science*, pages 105–116. Springer, 2011.

[6] J. A. Brzozowski and H. Tamm. Quotient complexities of atoms of regular languages. In H. Yen and O. H. Ibarra, editors, *Developments in Language Theory - 16th International Conference, DLT 2012, Taipei, Taiwan, August 14-17, 2012. Proceedings*, volume 7410 of *Lecture Notes in Computer Science*, pages 50–61. Springer, 2012.

[7] J. A. Brzozowski and H. Tamm. Complexity of atoms of regular languages. *International Journal of Foundations of Computer Science*, 24(07):1009–1027, 2013.

[8] R. L. Graham, D. E. Knuth, and O. Patashnik. *Concrete Mathematics: A Foundation for Computer Science*. Addison-Wesley, 1994.

[9] L. Schaeffer. Bounds for the atom complexity function. Unpublished, 2012.

Received 15th June 2015

Acta Cybernetica **22** (2015) 359–371.

A Novel Cryptosystem Based on Gluškov Product of Automata*†

Pál Dömösi‡ and Géza Horváth§

Abstract

The concept of Gluškov product was introduced by V. M. Gluškov in 1961. It was intensively studied by several scientists (first of all, by Ferenc Gécseg and the automata-theory school centred around him in Szeged, Hungary) since the middle of 60's. In spite of the large number of excellent publications, no application of Gluškov-type products of automata in cryptography has arisen so far. This paper is the first attempt in this direction.

Keywords: cryptosystem, Gluškov product of automata

1 Dedication

This paper is dedicated to the memory of our late colleague, teacher and friend, Professor Ferenc Gécseg who has been a central figure in modern automata theory. He established the world famous research school of Szeged University in automata theory. His death is an irreplaceable loss for the whole research community of theoretical computer science.

2 Introduction

The connection of certain automata through various communication links leads to the notion of composition of automata [9]. A substantial body of literature in this important scientific field has been published by researchers belonging to the automata-theory school centred around Ferenc Gécseg in Szeged, Hungary [8, 9]. The specific concept of automaton also applied in cryptography, the cellular

*In memory of Professor Ferenc Gécseg

†The second author was supported by the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project. The project has been supported by the European Union, co-financed by the European Social Fund.

‡Institute of Mathematics and Informatics, College of Nyíregyháza, H-4400 Nyíregyháza, Sóstói út 36, Hungary, E-mail: domosi@nyf.hu

§Faculty of Informatics, University of Debrecen, H-4028 Debrecen, Kassai út 26, Hungary, E-mail: horvath.geza@inf.unideb.hu

automaton, can also be regarded a special composition of automata, where the cells functioning as the members of the composition are composed of one and the same type of elementary automata, and the pattern of the communication links and connections between these elementary automata is a simple network. Despite the large number of publications on compositions of automata (authored predominantly by Hungarian researchers), no cryptographic applications of the results have been disclosed so far.

Several cryptosystems have been designed on the basis of abstract automata. Some of them are based on Mealy automata or their generalization (see, for exaple [1, 14, 20, 21]), some of them are based on cellular automata (see, for example [12, 13, 16, 23]), while [6] is based on automata without outputs . The best-known abstract automata based cryptosystems all share the common problem of serious realization difficulties: some systems are easy to defeat [2, 3, 4, 17, 19, 22], the technical realization of others result in slow performance [6, 7, 12, 21], and still others exhibit difficulties in the choice of the key-automaton [5, 16]. These drawbacks justify the need of novel cryptosystems overcoming these problems. By some experimental results we will show the security of the proposed system. (Serious security analysis should be necessary in the future work.) By an example we show that the technical realization of the novel system is not difficult. Moreover, we give a method to generate key automata easily.

A Gluškov product of automata [11] is loosely defined as a collection of automata that each of which changes its state at discrete time steps by a local transition function of the states and a global input. Moreover, the synchronous action of the local state transitions defines a global transition on the entire product. Thus a Gluškov product of automata is also an automaton. Usually it is assumed that the component automata are connected together according to a directed graph $\mathcal{D}$. The vertices of $\mathcal{D}$ are considered as automata and the edges indicate the existence of communication links. Thus $\mathcal{D}$ has no parallel edges.

An important observation of this paper is that, using the concept of Gluškov product, we can store certain properties of very large automata such that their transitions can be computed easily. By this observation, we can built new secure symmetric block ciphers based on Gluškov product of automata.

3 Preliminaries

We start with some standard concepts and notation. For all notions and notation not defined here we refer to the monographs [8, 9, 10, 15, 18]. A *word* (over Σ) is a finite sequence of elements of some nonempty and finite set Σ. We call the set Σ an *alphabet*, the elements of Σ *letters*. By the *free monoid* Σ^* *generated by* Σ we mean the set of all words (including the *empty word* λ) having catenation as multiplication. We set $\Sigma^+ = \Sigma^* \setminus \{\lambda\}$, where the subsemigroup Σ^+ of Σ^* is said to be the *free semigropu generated by* Σ. By an *automaton* we mean a deterministic finite automaton without outputs. In more details, an automaton is an algebraic structure $\mathcal{A} = (A, \Sigma, \delta)$ consisting of the nonempty and finite *state set* A, the

nonempty and finite *input set* Σ, and a *transition function* $\delta : A \times \Sigma \to A$. The elements of the state set are the *states*, and the elements of the input set are the *input signals*. An element of A^+ is called a *state word* [1] and an element of Σ^* is called an input word. State and input words are also called *state strings* and *input strings,* respectively. If a state string $a_1a_2 \cdots a_s$ $(a_1, \ldots, a_s \in A)$ has at least three elements, the states $a_2, a_3, \ldots, a_{s-1}$ are also called intermediate states. It is understood that δ is extended to $\delta^* : A \times \Sigma^* \to A^+$ with $\delta^*(a, \lambda) = a,\ \delta^*(a, xq) = \delta(a, x)\delta^*(\delta(a, x), q), a \in A, x \in \Sigma, q \in \Sigma^*$. In other words, $\delta^*(a, \lambda) = a$ and for every nonempty input word $x_1x_2 \cdots x_s \in \Sigma^+$ (where $x_1, x_2, \ldots, x_s \in \Sigma$) there are $a_1, \ldots, a_s \in A$ with $\delta(a, x_1) = a_1, \delta(a_1, x_2) = a_2, \ldots, \delta(a_{s-1}, x_s) = a_s$ such that $\delta^*(a, x_1 \cdots x_s) = a_1 \cdots a_s$.

In the sequel, we will consider the transition of an automaton in this extended form and thus we will denote it by the same Greek letter δ. If b is the last letter of $\delta(a, w)$ for some $a, b \in A, w \in \Sigma^*$ then we say that w *takes* the automaton from its state a into state b, and we also say that the automaton *goes* from state a into state b under the effect of w. The automaton $\mathcal{B} = (B, Y, \delta_{\mathcal{B}})$ with $B \subseteq A$, $Y \subseteq \Sigma$ and $\delta_{\mathcal{B}}(a, x) = \delta(a, x), a \in B, x \in Y$ is a *subautomaton* of $\mathcal{A}$. In particular, if $B \subseteq A$ and $Y = \Sigma$ then $\mathcal{B}$ is a state-subautomaton of $\mathcal{A}$. Moreover, if $B = A$ and $Y \subseteq \Sigma$ then $\mathcal{B}$ is an input-subautomaton of $\mathcal{A}$. The automaton $\mathcal{C} = (C, \Sigma_{\mathcal{C}}, \delta_C)$ is isomorphic to $\mathcal{A}$ if there are bijective mappings $\tau_1 : C \to A, \tau_2 : \Sigma_{\mathcal{C}} \to \Sigma$ with $\tau_1(\delta_C(c, x)) = \delta(\tau_1(c), \tau_2(x)), c \in C, x \in \Sigma_{\mathcal{C}}$. If $\Sigma_{\mathcal{C}} = \Sigma$ and $\tau_2(x) = x, x \in \Sigma$ then we say that $\mathcal{C}$ is state isomorphic to $\mathcal{A}$. In this case, we also say that $\mathcal{A}$ is a state-isomorphic copy of $\mathcal{C}$ and vice versa.[2]

The transition matrix of an automaton is a matrix with rows corresponding to each input and columns corresponding to each state; the state $\delta(a, x)$ is put at the entry of any row indicated by an input $x \in \Sigma$ and any column indicated by a state $a \in A$. If all rows of the transition matrix are permutations of the state set then we speak about a *permutation automaton.*

Next we prove the following statement.

Proposition 1. *Given a permutation automaton $\mathcal{A} = (A, \Sigma, \delta)$, for every pair $b \in A, x \in \Sigma$, there exists exactly one $a \in A$ with $\delta(a, x) = b$.*

Proof. Assume that there exists no $a \in A$ with $\delta(a, x) = b$. Then the row of of the transition matrix labeled by x does not contain b. But then $\mathcal{A}$ is not a permutation automaton, a contradiction.

Next we assume that there are $a_1, a_2 \in A$ with $a_1 \neq a_2$ $\delta(a_1, x) = b$ and $\delta(a_2, x) = b$. Then the row of of the transition matrix labeled by x contains b two times, a contradiction again. □

Let $\mathcal{A}_i = (A_i, \Sigma_i, \delta_i)$ be automata where $i \in \{1, \ldots, n\},\ n \geq 1$. Take a finite nonvoid set Σ and a *feedback function* $\varphi_i : A_1 \times \cdots \times A_n \times \Sigma \to \Sigma_i$ for every $i \in \{1, \ldots, n\}$. A *Gluškov-type product* of the automata $\mathcal{A}_i$ with respect

[1] The empty word is not considered as a state word.

[2] Obviously, then the bijective mapping $\tau_1 : C \to A$ unambigously determines the state isomorphism of $\mathcal{C}$ onto $\mathcal{A}$.

to the feedback functions φ_i ($i \in \{1, \ldots, n\}$) is defined to be the automaton $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n(\Sigma, (\varphi_1, \ldots, \varphi_n))$ with state set $A = A_1 \times \cdots \times A_n$, input set Σ, transition function δ given by $\delta((a_1, \ldots, a_n), x) = (\delta_1(a_1, \varphi_1(a_1, \ldots, a_n, x)), \ldots, \delta_n(a_n, \varphi_n(a_1, \ldots, a_n, x)))$ for all $(a_1, \ldots, a_n) \in A$ and $x \in \Sigma$.

We shall use the feedback functions $\varphi_i, i \in \{1, \ldots, n\}$ in an extended sense as mappings $\varphi_i^* : A_1 \times \cdots \times A_n \times \Sigma^* \to \Sigma_i^*$, where $\varphi_i^*(a_1, \ldots, a_n, \lambda) = \lambda$, and $\varphi_i^*(a_1, \ldots, a_n, px) = \varphi_i^*(a_1, \ldots, a_n, p)\varphi_i(\delta_1(a_1, \varphi_1^*(a_1, \ldots, a_n, p)), \ldots, \delta_n(a_n, \varphi_n^*(a_1, \ldots, a_n, p)), x)$, $a_i \in A_i$, $i \in \{1, \ldots, n\}, p \in \Sigma^*, x \in \Sigma$. In the sequel, $\varphi_i^*, i \in \{1, \ldots, n\}$ will also be denoted by φ_i.

We can imagine this structure as a working model in the following way. The product is a collection of automata so that every member of this collection is supplied with a transformer which is a special type of finite state transducer. The transformers, realizing the feedback functions mentioned above, are able to get an input vector containing a common external input sign and the state of all component automata. They can each transform this input vector into an appropriate input sign for their component automaton. The product is at work along a discrete time scale in the following way: all transformers of the product get a common external input sign x, and simultaneously, all transformers get the value of the instantaneous states $a_1, \ldots, a_n$ of all component-automata as input information. Induced by this this input vector $(a_1, \ldots, a_n, x)$, the transformers produce an input sign $x_i = \varphi_i(a_1, \ldots, a_n, x), i \in \{1, \ldots, n\}$ for their component-automata. Then, these (transformed) input signs take every component-automaton into a new (not necessarily different) state $\delta_i(a_i, x_i) = \delta_i(a_i, \varphi_i(a_1, \ldots, a_n, x))$, and then, in the next time period, the whole process takes place again. We will use several generalizations and several restrictions of this concept. If the transformers are able to produce not only single input signs but entire input words (strings of input signs), then induced by the inner input sign x and the value of the instantaneous states $a_1, \ldots, a_n$ they produce a (possibly empty) input word $\varphi_i(a_1, \ldots, a_n, x)$ working as microprocessors, for their component automata then we get the model of the *generalized product*.

If we assume that transformers do not necessarily have access to all the instantaneous states of component automata, but only some restricted subset, then we will get the models of several special types of the products [8, 9].

It is clear that, by definition, a Gluškov product is a parallel working system. Since parallel working Gluškov product is not appropriate for block cipher, we define its sequentially working version called *sequentially working Gluškov product*.

Consider the above defined Gluškov product modifying its transition function in the following way. Let δ be given by
$\delta((a_1, \ldots, a_n), x) = (\delta_1(a_1, \varphi_1(a_1, \ldots, a_n, x)), \delta_2(a_2, \varphi_2(a_1', a_2, \ldots, a_n, x)), \ldots, \delta_{n-1}(a_{n-1}, \varphi_{n-1}(a_1', \ldots, a_{n-2}', a_{n-1}, a_n, x)), \delta_n(a_n, \varphi_n(a_1', \ldots, a_{n-1}', a_n, x)))$ for all $(a_1, \ldots, a_n) \in A$ and $x \in \Sigma$, where, in order, $a_1' = \delta_1(a_1, \varphi_1(a_1, \ldots, a_n, x)), a_2' = \delta_2(a_2, \varphi_2(a_1', \ldots, a_n, x)), \ldots, a_{n-1}' = \delta_{n-1}(a_{n-1}, \varphi_{n-1}(a_1', \ldots, a_{n-2}', a_{n-1}, a_n, x))$.

Given a function $f : X_1 \times \cdots \times X_n \to Y$, we say that f is *really independent of its i-th variable* if for every pair $(x_1, \ldots, x_n), (x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_n) \in X_1 \times \cdots \times X_n$, $f(x_1, \ldots, x_n) = f(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_n)$. Otherwise we say

that f *really depends on its* i*-th variable.*

A (finite) *directed graph* (or, in short, a *digraph*) $\mathcal{D} = (V, E)$ (of order $n > 0$) is a pair consisting of sets of *vertices* $V = \{v_1, \ldots, v_n\}$ and *edges* $E \subseteq V \times V$. Elements of V are sometimes called *nodes*. If $|V| = n$ then we also say that $\mathcal{D}$ is a digraph of order n.

Given a digraph $\mathcal{D} = (V, E)$, we say that the above defined Gluškov product (sequentially working Gluškov product) is a $\mathcal{D}$-product (sequentially working $\mathcal{D}$-product) if for every pair $i, j \in \{1, \ldots, n\}$, $(i, j) \notin E$ implies that the feedback function φ_i is really independent of its j-th variable.

By a *key automaton* we mean a sequentially working Gluškov product having the following properties:

- it consists of automata components that are state isomorphic to each other so that their state sets also coincide with each other,
- it has the same state and input sets which are sets of all strings with a given length over a fixed alphabet,
- it is a permutation automaton.

4 Encryption and Decryption

Both of the encryption and decryption apparatus use the same key automaton and they use the same pseudorandom generator. We have to use the same pseudorandom blocks during the encryption and decryption processes, because otherwise decryption is impossible, and these pseudorandom blocks have to be secret, otherwise the system is vulnerable. Modern block ciphers create different ciphertext each time when they encrypt the same plaintext. To reach this goal, we have to change the seed of the pseudorandom generator each time when we use encryption. It is not too difficult to satisfy all these properties: we need two blocks, one is constant, secret and part of the key, let us call it ,,core vector", and the other block is changed each time when we use encryption, this one is public, – it is the first block of the ciphertext, – and let us call it ,,initialization vector". The recent seed can be calculated as a function of these two blocks. The most simple solution is to use the exclusive or (bitwise addition modulo 2) operator. In this way the seed will be secret, both of the encryption and decryption process calculate the same seed, they can calculate the same secret pseudorandom blocks, and the seed and the pseudorandom blocks are changed each time, when we use encryption.

There is a fixed positive integer k which is the number of the rounds (see later). Before the encryption procedure, the pseudorandom generator gets its initialization vector as a true random sign $r_1 \ldots r_n \in \Sigma^n$, where the pseudorandom alphabet Σ is also the plaintext and the ciphertext alphabet simultaneously. This initialization vector will be also the first block of the ciphertext.

The encryption procedure is the following. The apparatus reads the plaintext block-by-block and, after reading the next plaintext block $a_1 \cdots a_n \in \Sigma^n$ (first the first block), it generates the second, third, etc. blocks of the ciphertext in the following way.

First the random number generator generates a word $w_1 \cdots w_k$ of pseudorandom sequences, where $w_1, \ldots, w_k \in \Sigma^n$. The key automaton $\mathcal{A} = (\Sigma^n, \Sigma^n, \delta_\mathcal{A})$ goes from state $(a_1, \ldots, a_n)$ into state $(c_1, \cdots, c_n) = \delta_\mathcal{A}((a_1, \ldots, a_n), w_1 \cdots w_k)$, where $a_1 \cdots a_n$ is the referred next plaintext block. The state $(c_1, \ldots, c_n)$ will be performed sequentially such that, in order, we specify the state $\delta_\mathcal{A}((a_1, \ldots, a_n), w_1)$ by $(a_1, \ldots, a_n)$ and w_1, the state $\delta_\mathcal{A}((a_1, \ldots, a_n), w_1 w_2)$, by $\delta_\mathcal{A}((a_1, \ldots, a_n), w_1)$ and w_2, ..., the state $\delta_\mathcal{A}((a_1, \ldots, a_n), w_1 \cdots w_{k-1})$ by $\delta_\mathcal{A}((a_1, \ldots, a_n), w_1 \cdots w_{k-2})$ and w_{k-1}, the state $(c_1, \ldots, c_n) = \delta_\mathcal{A}((a_1, \ldots, a_n), w_1 \cdots w_k)$ by $\delta_\mathcal{A}((a_1, \ldots, a_n), w_1 \cdots w_{k-1})$ and w_k.

Let $w_i = (x_1, \ldots, x_n)$ where $x_1, \ldots, x_n \in \Sigma$ for some $i \in \{1, \ldots, k\}$ and let us define $(d_1, \ldots, d_n)$ and $(e_1, \ldots, e_n)$ by $(e_1, \ldots, e_n) = \delta_\mathcal{A}((a_1, \ldots, a_n), w_1 \cdots w_i)$ and $(d_1, \ldots, d_n) = \delta_\mathcal{A}((a_1, \ldots, a_n), w_1 \cdots w_{i-1})$ if $i > 1$, moreover, $(e_1, \ldots, e_n) = \delta_\mathcal{A}((a_1, \ldots, a_n), w_1)$ and $(d_1, \ldots, d_n) = (a_1, \ldots, a_n)$ if $i = 1$.

Clearly, then $(e_1, \ldots, e_n) = \delta_\mathcal{A}((d_1, \ldots, d_n), (x_1, \ldots, x_n))$.

This transition will be performed sequentially in the following way.

$e_1 = \delta_1(d_1, \varphi_1(d_1, d_2, \ldots, d_n, (x_1, \ldots, x_n))$,
$e_2 = \delta_2(d_2, \varphi_2(e_1, d_2, d_3, \ldots, d_n, (x_1, \ldots, x_n))$,
...
$e_{n-1} = \delta_{n-1}(d_{n-1}, \varphi_{n-1}(e_1, \ldots, e_{n-2}, d_{n-1}, d_n, (x_1, \ldots, x_n))$,
$e_n = \delta_n(d_n, \varphi_n(e_1, \ldots, e_{n-1}, d_n, (x_1, \ldots, x_n))$.

Applying the above procedure in k round, we finally receive the state $(c_1, \ldots, c_n)$. Then, concatenating the calculated blocks, we will get the ciphertext $c_1 \cdots c_n$.

The decryption procedure is the following. Before the decryption procedure, the pseudorandom generator gets the first ciphertext block as its initialization vector $r_1 \ldots r_n \in \Sigma^n$.

Then the apparatus reads the ciphertext block-by-block and, after reading the next ciphertext block $c_1 \cdots c_n \in \Sigma^n$ (first the second block), it generates the first, second, third, etc. blocks of the plaintext back in the following way.

First the random number generator generates the same word $w_1 \cdots w_k$ of pseudorandom sequences as at the encryption. Recall that the key automaton is a permutation automaton. Therefore, by Proposition 1, it has exactly one state $(a_1, \ldots, a_n)$ from which the key automaton goes into the state $(c_1, \ldots, c_n)$ under the effect of $w_1 \cdots w_k$. Then, applying the transition $(c_1, \cdots, c_n) = \delta_\mathcal{A}((a_1, \ldots, a_n), w_1 \cdots w_k)$ the plaintext block $a_1 \cdots a_k$ can be unambiguously recovered.

We specify the state $\delta_\mathcal{A}((a_1, \ldots, a_n), w_1 \cdots w_{k-1})$ by $(c_1, \ldots, c_n) = \delta_\mathcal{A}((a_1, \ldots, a_n), w_1 \cdots w_k)$ and w_k, the state $\delta_\mathcal{A}((a_1, \ldots, a_n), w_1 \cdots w_{k-2})$ by $\delta_\mathcal{A}((a_1, \ldots, a_n), w_1 \cdots w_{k-1})$ and w_{k-1}, ..., the state $\delta_\mathcal{A}((a_1, \ldots, a_n), w_1)$ by $\delta_\mathcal{A}((a_1, \ldots, a_n), w_1 w_2)$ and w_2, the state $(a_1, \ldots, a_n)$ by $\delta_\mathcal{A}((a_1, \ldots, a_n), w_1)$ and w_1.

The vectors $w_i, (d_1, \ldots, d_n)$, and $(e_1, \ldots, e_n)$ are defined in the same way as it is done at the encryption procedure. In more details, similarly as previously, let $w_i = (x_1, \ldots, x_n)$ where $x_1, \ldots, x_n \in \Sigma$ for some $i \in \{1, \ldots, k\}$ and let us define

$(d_1, \ldots, d_n)$ and $(e_1, \ldots, e_n)$ by $(e_1, \ldots, e_n) = \delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1 \cdots w_i)$ and $(d_1, \ldots, d_n) = \delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1 \cdots w_{i-1})$ if $i > 1$, moreover, $(e_1, \ldots, e_n) = \delta_{\mathcal{A}}((a_1, \ldots, a_n), w_1)$ and $(d_1, \ldots, d_n) = (a_1, \ldots, a_n)$ if $i = 1$.

To recover $d_1 \cdots d_n$, the following equalities are used.

By $e_n = \delta_n(d_n, \varphi_n(e_1, \ldots, e_{n-1}, d_n, (x_1, \ldots, x_n))$, we can determine d_n,

by $e_{n-1} = \delta_{n-1}(d_{n-1}, \varphi_{n-1}(e_1, \ldots, e_{n-2}, d_{n-1}, d_n, (x_1, \ldots, x_n))$, we can determine d_{n-1},

$\ldots$,

by $e_2 = \delta_2(d_2, \varphi_2(e_1, d_2, \ldots, d_n, (x_1, \ldots, x_n))$,we can determine d_2,

by $e_1 = \delta_1(d_1, \varphi_1(d_1, d_2, \ldots, d_n, (x_1, \ldots, x_n))$,we can determine d_1.

Thus we can get the plaintext block in k rounds back.

Therefore, if all of $\varphi_1, \ldots, \varphi_n$ can be computed easily, then the proposed system could be effective.

To sum up, the discussed cryptosystem is a block cipher. Since the key automaton is a permutation automaton, for every ciphertext there exists exactly one plaintext making the encryption and decryption unambiguous. Moreover, there is a huge number of corresponding encoded messages to each plaintext so that several encryptions of the same plaintext yield several distinct ciphertexts.

5 Example

Next we consider a special key automaton for which the proposed cryptosystem is effective and secure. We are going to use a sequentially working $\mathcal{D}$-product of automata for *key automaton* in this Section.

Let Σ be the set of all binary strings with a given length $\ell \geq 1$ and let n be a positive integer.

Let $\mathcal{A}_1 = (\Sigma, \Sigma \times \Sigma, \delta_{\mathcal{A}_1})$ be a permutation automaton and let $\mathcal{A}_i = (\Sigma, \Sigma \times \Sigma, \delta_{\mathcal{A}_i}), i = 2, \ldots, n$ be state-isomorphic copies of $\mathcal{A}_1$ such that $\mathcal{A}_1, \ldots, \mathcal{A}_n$ are pairwise distinct.[3] Given a digraph $\mathcal{D} = (V, E)$ with $V = \{1, \ldots, n\}, E = \{(n, 1), (1, 2), \ldots, (n-1, n)\}$ define the Gluškov-type product, called $\mathcal{D}$-product, $\mathcal{A}_{\mathcal{D}} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n(\Sigma^n, (\varphi_1, \ldots, \varphi_n))$ of $\mathcal{A}_1, \ldots, \mathcal{A}_n$ so that for every $(a_1, \ldots, a_n)$, $(x_1, \ldots, x_n) \in \Sigma^n, i \in \{1, \ldots, n\}$,

$\varphi_1(a_1, \ldots, a_n, (x_1, \ldots, x_n)) = (a_n \oplus x_n, x_1)$, where $a_n \oplus x_n$ is the bitwise addition modulo 2 of a_n and x_n,

$\varphi_i(a_1, \ldots, a_n, (x_1, \ldots, x_n)) = (a_{i-1} \oplus x_{i-1}, x_i), i = 2, \ldots, n$ where $a_{i-1} \oplus x_{i-1}$ is the bitwise addition modulo 2 of a_{i-1} and x_{i-1}.

Then the sequentially working version of $\mathcal{A}_{\mathcal{D}}$ is the automaton $\mathcal{B} = (\Sigma^n, \Sigma^n, \delta_{\mathcal{B}})$, where for every $(a_1, \ldots, a_n), (x_1, \ldots, x_n) \in \Sigma^n$, $\delta_{\mathcal{B}}((a_1, \ldots, a_n), (x_1, \ldots, x_n)) = (b_1, \ldots, b_n)$ such that

$b_1 = \delta_{\mathcal{A}_1}(a_1, \varphi_1(a_1, \ldots, a_n, (x_1, \ldots, x_n))$

and $\varphi_1(a_1, \ldots, a_n, (x_1, \ldots, x_n)) = (a_n \oplus x_n, x_1)$,

[3] In other words, for every $i, j \in \{1, \ldots, n\}$, $i \neq j$ implies $\mathcal{A}_i \neq \mathcal{A}_j$.

$b_2 = \delta_{\mathcal{A}_2}(a_2, \varphi_2(b_1, a_2, \ldots, a_n, (x_1, \ldots, x_n))$,
and $\varphi_2(b_1, a_2, \ldots, a_n, (x_1, \ldots, x_n)) = (b_1 \oplus x_1, x_2)$,
...
$b_{n-1} = \delta_{\mathcal{A}_{n-1}}(a_{n-1}, \varphi_{n-1}(b_1, \ldots, b_{n-2}, a_{n-1}, a_n, (x_1, \ldots, x_n))$,
and $\varphi_{n-1}(b_1, \ldots, b_{n-2}, a_{n-1}, a_n, (x_1, \ldots, x_n)) = (b_{n-2} \oplus x_{n-2}, x_{n-1})$,
$b_n = \delta_{\mathcal{A}_n}(a_n, \varphi_n(b_1, \ldots, b_{n-1}, a_n, (x_1, \ldots, x_n))$,
and $\varphi_n(b_1, \ldots, b_{n-1}, a_n, (x_1, \ldots, x_n)) = (b_{n-1} \oplus x_{n-1}, x_n)$.

Of course, the values of the feedback functions can be computed easily. By the encryption procedure, using the transition matrices of the component automata, we can specify easily the state b_1 from a_1, a_n, x_n, x_1, the state b_2 from a_2, b_1, x_1, x_2, ..., the state b_{n-1} from $a_{n-1}, b_{n-2}, x_{n-2}, x_{n-1}$, the state b_n from $a_n, b_{n-1}, x_{n-1}, x_n$. On the other hand, all component automata of the key automaton are permutation automata. Therefore, by the decryption procedure, using again the transition matrices of the component-automata, we can specify unambiguously the state a_n from $b_{n-1}, b_n, x_{n-1}, x_n$, the state a_{n-1} from $b_{n-2}, b_{n-1}, x_{n-2}, x_{n-1}$, ..., the state a_2 from b_1, b_2, x_1, x_2, the state a_1 from a_n, b_1, x_n, x_1.

6 Avalanche Effect

The avalanche effect is a very important property of block ciphers. We say the block cipher has avalanche effect when a small change in the plaintext block (or in the key) results a significant change in the corresponding ciphertext block, and also small change in the ciphertext block (or in the key) results a significant change in the corresponding plaintext block after decoding. In section 4 we introduced a very simple key automaton, which works well, but it has just limited avalanche effect. Suppose we have a plaintext block $a = (a_1, \ldots, a_n) \in \Sigma^n$, a pseudorandom block $w_1 = (x_1, \ldots, x_n) \in \Sigma^n$ and the key automaton $\mathcal{B} = (\Sigma^n, \Sigma^n, \delta_{\mathcal{B}})$ goes to the ciphertext block $b = (b_1, \ldots, b_n) \in \Sigma^n$ from a by the effect of w_1. (In short, $\delta_{\mathcal{B}}(a, w_1) = b$.) Let us define $c = (a_1, \ldots, a_{i-1}, c_i, a_{i+1}, \ldots, a_n) \in \Sigma^n$, where $a_i \neq c_i$, $1 < i < n$, and calculate the $d = \delta_{\mathcal{B}}(c, w_1)$ value. We will see that d starts with $b_1, \ldots, b_{i-1}$ so changing a_i to c_i has no effect for the first $i-1$ part of the ciphertext block. However, from the i-th part, we have appropriate avalanche effect. This is the same with the pseudorandom block, changing x_i to c_i ($x_i \neq c_i$, $1 < i < n$) has no effect for the first $i-1$ part of the ciphertext block, but it has appropriate avalanche effect from the i-th part of the ciphertext. The solution is simple. We should repeat the encoding procedure twice. First calculate the $a' = \delta_{\mathcal{B}}(a, w_1)$ block, then calculate the $b = \delta_{\mathcal{B}}(a', w_1)$ ciphertext block.

Unfortunately, the situation during the decoding is worst. Suppose we have the $b = (b_1, \ldots, b_n) \in \Sigma^n$ ciphertext block, the $w_1 = (x_1, \ldots, x_n) \in \Sigma^n$ pseudorandom block and the key automaton $\mathcal{B} = (\Sigma^n, \Sigma^n, \delta_{\mathcal{B}})$ goes to the ciphertext block b from the paintext block $a = (a_1, \ldots, a_n) \in \Sigma^n$ by the effect of w_1. (In short, $\delta_{\mathcal{B}}(a, w_1) = b$.) Let us define $\gamma_{\mathcal{B}}$ such that $\gamma_{\mathcal{B}}(\delta_{\mathcal{B}}(a, w), w) = a$ for each $a, w \in \Sigma^n$. In this case $\gamma_{\mathcal{B}}(b, w_1) = a$. Now let us define the $d = (b_1, \ldots, b_{i-1}, d_i, b_{i+1}, \ldots, b_n) \in \Sigma^n$, where

$b_i \neq d_i$, $1 < i \leq n$. Comparing a and $\gamma_{\mathcal{B}}(d, w_1)$ we can recognize that changing the i-th part of the ciphertext block has effect only on the i-th and $i-1$-th part of the plaintext block. This means we can not have appropriate avalanche effect during decoding using only the above defined $\gamma_{\mathcal{B}}$ function. To solve this problem, we have to use the $\delta_{\mathcal{B}}$ function twice during the decoding process.

Finally, we created the following function, which has 3 parameters, can do the encoding and the decoding, and – based on experimental results, – it has appropriate avalanche effect during the encoding and the decoding process:

$$f(a, w_1, w_2) = \gamma_{\mathcal{B}}(\gamma_{\mathcal{B}}(\delta_{\mathcal{B}}(\delta_{\mathcal{B}}(a, w_1), w_1), w_2), w_2).$$

This function first receives the plaintext block a and two pseudorandom blocks w_1 and w_2.

Then, it calculates the $a' = \delta_{\mathcal{B}}(a, w_1)$ value.

In the next round, it calculates the $a'' = \delta_{\mathcal{B}}(a', w_1)$ value.

In the next round, it calculates the $a''' = \gamma_{\mathcal{B}}(a'', w_2)$ value.

In the next round, it calculates the $b = \gamma_{\mathcal{B}}(a''', w_2)$ value, which is the ciphertext block.

Decoding done with the same function, but it has different parameters: $f(b, w_2, w_1)$. In this case the same f function first receives the ciphertext block b and the two pseudorandom blocks w_2 and w_1 in the opposite order.

Then, it calculates the $a''' = \delta_{\mathcal{B}}(b, w_2)$ value.

In the next round, it calculates the $a'' = \delta_{\mathcal{B}}(a''', w_2)$ value.

In the next round, it calculates the $a' = \gamma_{\mathcal{B}}(a'', w_1)$ value.

In the next round, it calculates the $a = \gamma_{\mathcal{B}}(a', w_1)$ value, which is the plaintext block.

For protection against chosen ciphertext attack, we recommend to repeat this procedure at least twice during the encoding and decoding process, with different pseudorandom numbers. For example, the ciphertext block b can be calculated from the plaintext block a by the function $f(f(a, w_1, w_2), w_3, w_4)$, with four pseudorandom number blocks w_1, w_2, w_3, w_4, and then, we can decipher the plaintext block a from the ciphertext block b using the function $f(f(b, w_4, w_3), w_2, w_1)$.

7 Experimental Results

We have been developed some practical tests using 16 bytes (128 bits) long input blocks, output blocks and pseudorandom blocks. It has been done for the cases when both of the encryption and decryption algorithms in Chapter 4 have been modified as it is formulated in Chapter 6.

7.1 Keyspace Size

Using the above mentioned parameters with 256 possible states, (1 byte long states,) we need 16 automata, having a transition matrix $2^{16} = 65536$ lines and $2^8 = 256$

columns. Each cell of the automaton contains 1 byte long data. (One state.) The size of the matrix is 16 megabytes, and the number of possible matrices is $256!^{65536}$, where the exclamation mark means the factorial operation. This is much more than good enough protection against brute-force attack. When we use isomorphic automata, this huge number should be further increase to have $256!^{65536} * 256!^{15} = 256!^{65551}$ possible keys.

7.2 Speed Test Results

The practical tests of the encoding and decoding algorithm were done on an average table PC, (3,1 GHz Intel Core I3-2100 processor, 4 Gigabyte RAM). The program we used was a well written C# implementation. The results of the speed tests of the 8 bit version can be seen in the table 1.

Table 1: Results of the speed tests

size (bytes)	encoding time	decoding time	encoded bytes per second
131104	00.0169140	00.0164919	7751212
524336	00.0572925	00.0573531	9151913
1048656	00.1111786	00.1098338	9432175
33556496	03.8841316	04.0200288	8639382
134225936	16.0446227	16.1320934	8365789

The results of the speed tests show that using an average PC, the encoding time is more than 7 megabytes per second, and decoding time is about the same.

7.3 Effectiveness of the Avalanche Effect

We used to test the avalanche effect in the following way. We chose 1000000 random plaintext blocks, encoded them, and then we changed 1 bit in each plaintext block, encoded again, then we calculated the number of the different bytes in the ciphertext blocks pair-wise. The opposite case has been also tested, namely there were chosen 1000000 random ciphertext blocks, we decoded them, and then we changed 1 bit in each ciphertext block, decoded again, and calculated the number of the different bytes in each plaintext blocks pair-wise. The results can be seen in the table 2.

Table 2: Results of the avalanche effect of encoding and decoding

different characters in one block	encoding	decoding
0-12	0	0
13	24	32
14	1771	1743
15	58851	59028
16	939354	939197

When we change only one bit in the plaintext block, the difference between the corresponding ciphertext blocks will be really huge in the majority of the cases. The same effect can be seen in the opposite case, changing one bit in the ciphertext block results huge difference in the plaintext block as well.

We created another table as well. In this table we calculated the optimal avalanche effect. We had choosen 2×1000000 completely random blocks, and then calculated the difference between them pair-wise. The results can be seen in the table 3.

Table 3: Results of the avalanche effect of complete random blocks

different characters in one block	
0-12	0
13	32
14	1693
15	58681
16	939594

By our experimental results, we can conclude that the algorithm has the optimal avalanche effect, and an appropriate speed (more than 7 megbyte/s). Of course the speed of the algorithm depends on the hardware and the programming language / program code as well.

8 Conclusion and Future Work

This paper is devoted to propose a novel cryptosystem based on Gluškov product of automata. By a simple example, its utility is shown. The avalanche effect tests show good results. Moreover, some experimental results show the effectiveness. However, serious security analysis and rigorous machine-independent investigation should be necessary in the future work.

References

[1] Atanasiu, A. A class of coders based on gsm. Acta Informatica, 29 (1992), 779-791.

[2] Bao, F. Cryptoanalysis of partially known cellular automata. IEEE Trans. on Computers, 53 (2004), 1493-1497.

[3] Bao, F. and Igarashi, Y. Break finite automata public key cryptosystems. In: Fülöp, Z., Gécseg F., eds., Proc. 22nd Int. Coll. On Automata Languages and Programming - ICALP'95, Szeged, Hungary, July 10-14, 1995, LNC 944, Springer-Verlag, Berlin, 1995, 147-158.

[4] Biham, E. Cryptoanalysis of the chaotic map cryptosystem suggested at EUROCRYPT'91. In: Davies, D. W., ed., Proc. Conf. Advances in Cryptology - EUROCRYPT'91, Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, LNCS 547 Springer-Verlag, Berlin, 1991, 532-534.

[5] Clarridge, A., Salomaa, K. A Cryptosystem Based on the Composition of Reversible Cellular Automata. In: Dediu, A.-H., Ionescu, A.-M., Martn-Vide, C., eds., Language and Automata Theory and Applications, Third International Conference, LATA 2009, Tarragona, Spain, April 2-8, 2009. Proceedings. Volume 5457 of Lecture Notes in Computer Science, pages 314-325, Springer, 2009.

[6] Dömösi, P. A Novel Cryptosystem Based on Finite Automata without Outputs. In: Ito, M., Kobayashi, Y., Shoji, Kunitaka, S., eds., AFLAS'08, Proc. Int. Conf. On Automata, Formal languages and Algebraic Systems, Kyoto, Japan, 20-22 September 2008, World Scientific, New Jersey, London, Singapore, Beijing, Shanghai, Hong Kong, Taipei, Chennai, 2010, 23-32.

[7] Dömösi, P. A novel stream cipher based on finite automata. Cryptosystem Based on Finite Automata without Outputs. In: Vlad, M. S. and Sgarciu, V., eds., Intellisec - The 1st International Workshop on Intelligent Security Systems, 11-14 November, 2009, Printech, Bucharest, Romania, 2009, 16-25.

[8] Dömösi, P. and Nehaniv, C. L. Algebraic theory of automata networks. An introduction. SIAM Monographs on Discrete Mathematics and Applications, 11. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2005.

[9] Gécseg, F. Products of Automata. EATCS Monogr. Theoret. Comput. Sci. 7, Springer-Verlag, Berlin, Heidelberg, New York, Tokyo, 1986.

[10] Gécseg, F. and Peák, I. Algebraic theory of automata. Akadémiai Kiadó Budapest (Hungary), 1972.

[11] Gluskov, V. M. Abstract theory of automata (in Russian). Uspekhi Mat. Nauk, 16 (101) (1961), 3-62; correction, ibid., 17 (104) (1962), 270.

[12] Guan, P. Cellular automaton public key cryptosystem. Complex Systems, 1 (1987), 51-56.

[13] Gutowitz, H. A. Method and Apparatus for Encryption, Decryption, and Authentication Using Dynamical Systems. US P 5,365,589, 1994.

[14] Gysin, M. One-key cryptosystem based on a finite non-linear automaton. Dawson, E., Golic,J., eds., Proc. Int. Conf. Proceedings of the Cryptography: Policy and Algorithms, CPAC95, Brisbane, Queensland, Australia, July 3-5, 1995. Lecture Notes in Computer Science 1029 Springer-Verlag, Berlin, 1995, 165-163.

[15] Hopcroft, J.E., Motwani, R., and Ullman, J. D. Introduction to Automata Theory (second edition). Addison-Wesley Series in Computer Science, Addison-Wesley Co., Reading, MA, 2001.

[16] Kari, J. Cryptosystems based on reversible cellular automata. University of Turku, Finland, April, 1992, preprint.

[17] Meier, W. and Staffelbach, O. Analysis of pseudo-random sequences generated by cellular automata. In: Davies, D. W., ed., Proc. Conf. Advances in Cryptology EUROCRYPT91, Workshop on the Theory and Application of Cryptographic Techniques, Brighton, UK, April 8-11, 1991, LNCS 547 Springer-Verlag, Berlin, 1991, 186-199.

[18] Menezes, A. J., Oorschot, P. C., Vanstone, S. A. Handbook of Applied Cryptography. CRC Press Series on Discrete Mathematics and Its Applications, CRC Press LLC, Boca Raton, FL, USA, 1996, 2001, 2008.

[19] Meskaten, T. On finite automaton public key cryptosystems. TUCS Technical Report No. 408, Turku Centre for Computer Science, Turku, 2001, 1-42.

[20] Rayward-Smith, V. J. Mealy machines as coding devices. In: H. J. Beker and F. C. Piper, eds., Cryptography and Coding, Claredon Press, Oxford, 1989.

[21] Tao, R. Finite Automata and Application to Cryptography. Springer-Verlag, Berlin, 2009.

[22] Wichmann, P. Cryptoanalysis of a modified rotor machine. In: Quisquarter, J.-J., Vandewalle, J., eds., Proc. Conf. Advances in Cryptology - EUROCRYPT'89, Workshop on the Theory and Applications of Cryptographic Techniques, Houthalen, Belgium, April 10-13, 1989, LNCS 434, Springer-Verlag, Berlin, 1990, 395-402.

[23] Wolfram, S. Cryptography with Cellular Automata. In: Hugh, C. W., ed., Proc. Conf. Advances in Cryptology - CRYPTO'85, Santa Barbara, California, USA, August 18-22, 1985, LNCS 218, Springer-Verlag, Berlin, 1986, 429-432.

Received 13th January 2015

Acta Cybernetica **22** (2015) 373–392.

Context-Free Tree Grammars are as Powerful as Context-Free Jungle Grammars

Frank Drewes* and Joost Engelfriet†

Dedicated to the memory of Ferenc Gécseg

Abstract

Jungles generalize trees by sharing subtrees and allowing garbage. It is shown that IO context-free tree grammars generate the same jungle languages as context-free jungle grammars. Also, they define the same subsets of any algebra.

Keywords: context-free tree grammar, jungle, delegation network

1 Introduction

One of the main motivations for studying tree language theory is that a tree over a ranked alphabet is a term, which can be interpreted as an element of any algebra, see, e.g., Sections I.2, I.3, II.1 and II.2 of the influential book of Gécseg and Steinby [13]. Thus, the interpretation of a tree language, i.e., a set of trees, becomes a subset of the algebra. Regular tree grammars [13, Section II.3] and context-free tree grammars [14, Section 15] generate tree languages. However, as shown by Mezei and Wright in [17], a regular tree grammar can also naturally be viewed as a system of equations (or, more informally, as a recursive program) that has a least fixed point semantics in any algebra, and thus defines a subset of the algebra. The main result of [17] is that, for any algebra, the semantics of a regular tree grammar G equals the interpretation of the tree language $L(G)$ generated by G. Thus, the semantics of the program G is determined by the set of syntactic objects it generates. This program-schematic result was generalized to context-free tree grammars in [11], both for call by value semantics vs. inside-out (IO) generation, and for call by name semantics vs. outside-in (OI) generation. However, in the call by value case it holds for deterministic algebras, but *not for nondeterministic algebras*. In a usual, deterministic algebra, each operator symbol of rank k

*Department of Computing Science, Umeå University, S–901 87 Umeå, Sweden, E-mail: drewes@cs.umu.se

†Leiden Institute of Advanced Computer Science, Leiden University, P.O. Box 9612, NL–2300 RA Leiden, The Netherlands, E-mail: j.engelfriet@liacs.leidenuniv.nl

is interpreted as a k-ary operation on the domain of the algebra, whereas in a nondeterministic algebra, it is interpreted as a $(k+1)$-ary relation (see, e.g., [13, Section II.2]). Thus, in a nondeterministic algebra, a tree is interpreted as a subset of the algebra; as in the deterministic case, a tree language is also interpreted as a subset of the algebra, viz. the union of the interpretations of its elements. Since grammars are essentially nondeterministic programs, it is natural to interpret them in nondeterministic algebras.

It was shown in [5] that the call by value case for nondeterministic algebras can be handled by considering *jungles* (or DOAGs, directed ordered acyclic graphs) instead of trees. A jungle is a representation of a tree, in which equal subtrees can be shared, and in which "garbage" can occur that is not used in the tree, see, e.g., [1, 15, 16, 18].[1] Jungles can be interpreted in any nondeterministic algebra, in a natural way. The sharing of subtrees allows to fix a nondeterministic choice for later multiple use, whereas the garbage allows to force the evaluation of trees that are later disregarded. As shown in [5], a context-free tree grammar G can be turned into a graph grammar that generates jungles, in a straightforward way, such that the call by value semantics of G equals the interpretation of the "jungle language" $L_{\mathrm{J}}(G)$ generated by G, for any nondeterministic algebra.

On the basis of the above "Mezei-and-Wright-like" result for $L_{\mathrm{J}}(G)$, one may ask whether context-free tree grammars have the same jungle generating power as *context-free jungle grammars*, which are context-free graph grammars in which all right-hand sides of rules are jungles (see [5, Definition 7.6]). In this paper, we answer this question affirmatively. Moreover, we define the least fixed point semantics of a context-free jungle grammar in any nondeterministic algebra, viewing the grammar as a system of equations, and we prove that context-free jungle grammars define the same subsets of the algebra as IO context-free tree grammars. As a corollary we obtain that the above Mezei-and-Wright-like result also holds for context-free jungle grammars G. Finally, a context-free jungle grammar generates the tree language obtained by unfolding the generated jungles, and we show that context-free jungle grammars generate the same tree languages as IO context-free tree grammars.

Thus we conclude that, in all respects, IO context-free tree grammars have the same power as context-free jungle grammars.

In Section 2 we define basic concepts, such as trees and IO context-free tree grammars. Jungles are defined in Section 3, and we define the substitution of one jungle for a node of another jungle. It is shown that this substitution is confluent and associative (in the sense of [3]), a folklore result. In Section 4 we define context-free jungle grammars (CFJGs), in such a way that context-free tree grammars (CFTGs) are a special case. The derivations of a CFJG use the jungle substitution defined in the previous section. We show the simple fact that the rules of a CFJG can be substituted into one another (generalizing the corresponding property of context-free string grammars), and we prove our main result: for every CFJG G there is a CFTG H that generates the same jungle language. As a corollary we obtain that CFJGs generate the same tree languages as IO CFTGs. In

[1] When trees are called terms, jungles are called term graphs.

Section 5 we turn to semantics. We recall (nondeterministic) algebras and define the interpretation of jungles in such an algebra. Then we introduce the notion of a *jungle delegation network*, which is a CFJG G together with an algebra A. It generalizes the (finitary, tree) delegation network of [4, 5], which is an IO CFTG with an algebra. Finally, we define the least fixed point semantics of a jungle delegation network (G, A), and prove that it defines the same subset of A as the tree delegation network (H, A), where H is as above. As a corollary we obtain that that subset is equal to the interpretation in A of the jungle language $L_{\mathrm{J}}(G)$ generated by G: our Mezei-and-Wright-like result for context-free jungle grammars.

The results of this paper were already suggested in the Conclusion of [5].

2 Basic Terminology

The set of all natural numbers (including zero) is denoted $\mathbb{N}$. For $n \in \mathbb{N}$, we let $[n] = \{1, \ldots, n\}$. The set of all finite strings (or sequences) over a set A is denoted by A^*, and λ denotes the empty string. The length of a string u is denoted by $|u|$.

We assume functions to be total, i.e., if $f\colon A \to B$ is a function, then $f(a)$ is defined for every $a \in A$. Functions from A to B are a special case of binary relations $r \subseteq A \times B$. As usual, we let $r(A') = \{b \in B \mid \exists a \in A'\colon (a, b) \in r\}$ for $A' \subseteq A$, and $r(a) = r(\{a\})$ for $a \in A$. Note that, in this way, r can be viewed as a "nondeterministic function" $r\colon A \to \mathcal{P}(B)$ where $\mathcal{P}(B)$ is the powerset of B.

A *signature* (or ranked alphabet) is a pair (Σ, rk), where Σ is a finite set of symbols, and rk assigns to every $\mathtt{f} \in \Sigma$ a *rank* $rk(\mathtt{f}) \in \mathbb{N}$. We will denote (Σ, rk) simply by Σ. If necessary, the rank k of a symbol $\mathtt{f}$ is indicated by writing $\mathtt{f}$ as $\mathtt{f}^{(k)}$.

The set of all *trees over* Σ is denoted by T_Σ. It is the smallest set of strings such that for all $k \in \mathbb{N}$, $\mathtt{f}^{(k)}$ in Σ, and $t_1, \ldots, t_k \in \mathrm{T}_\Sigma$, the string $\mathtt{f}(t_1, \ldots, t_k)$ is in T_Σ (where the parentheses and the comma are assumed to be special symbols not in Σ). A tree of the form $\mathtt{f}(\,)$, where $\mathtt{f}$ has rank 0, is identified with the string $\mathtt{f}$ of length 1. A subset of T_Σ is a *tree language*.

As usual, a tree $t \in \mathrm{T}_\Sigma$ will be identified with a graph whose nodes are labelled with symbols in Σ. A node is a string in $(\mathbb{N} \setminus \{0\})^*$ which, intuitively, represents the Dewey path from the root to the node. Thus, λ is the root of t and vi is the i-th child of node v. Formally, we define the set $V(t)$ of *nodes* of t, the *subtree* t/v at a node v, and the *label* $\ell_t(v)$ of node v inductively, as follows. If $t = \mathtt{f}(t_1, \ldots, t_k)$, then $V(t) = \{\lambda\} \cup \{iv \mid i \in [k], v \in V(t_i)\}$; furthermore, $t/\lambda = t$, $\ell_t(\lambda) = \mathtt{f}$, and, for all $i \in [k]$ and $v \in V(t_i)$, $t/iv = t_i/v$ and $\ell_t(iv) = \ell_{t_i}(v)$. A node v of t is said to be an occurrence of the symbol $\ell_t(v)$.

As usual, to define the substitution of a tree s for a node v of a tree t, we use the set of *variables* $X = \{x_1, x_2, x_3, \ldots\}$. For $k \in \mathbb{N}$, $X_k = \{x_1, \ldots, x_k\}$ is a signature such that x_i has rank 0 for every $i \in [k]$. We assume X to be disjoint with all the usual signatures. For such a signature Σ, the set $\mathrm{T}_{\Sigma \cup X_k}$ is denoted by $\mathrm{T}_\Sigma(X_k)$; it is the set of *trees with k variables*.

For $t \in \mathrm{T}_\Sigma$, $v \in V(t)$ and $s \in \mathrm{T}_\Sigma(X_k)$, where $k = rk(\ell_t(v))$, the *substitution* of s for v in t, denoted $t[v \leftarrow s]$, is defined as follows:

- $t[\lambda \leftarrow s]$ is the result of substituting t/i for each occurrence of x_i in s;
- if $t = \mathtt{f}(t_1, \ldots, t_m)$, then $t[iv \leftarrow s] = \mathtt{f}(t_1, \ldots, t_{i-1}, t_i[v \leftarrow s], t_{i+1}, \ldots, t_m)$.

This notion of substitution leads to the definition of context-free tree grammars (see, e.g., [14, Section 15]).

Definition 1. *A* context-free tree grammar *(abbreviated CFTG) is a four-tuple* $G = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$ *such that*

- Ξ *and* Σ *are disjoint signatures of* nonterminals *and* terminals*, respectively,*
- R *is a finite set of* rules *of the form* $\mathtt{g}(x_1, \ldots, x_k) \to s$*, where* $k \in \mathbb{N}$*,* $\mathtt{g}^{(k)} \in \Xi$ *and* $s \in \mathrm{T}_{\Xi \cup \Sigma}(X_k)$*, and*
- $\mathtt{g}_{\mathrm{in}} \in \Xi$ *is the* initial nonterminal*, of rank* 0.

For trees $t, t' \in \mathrm{T}_{\Xi \cup \Sigma}$*, there is an IO derivation step* $t \Rightarrow_{G,\mathrm{IO}} t'$ *if there are a node* $v \in V(t)$ *and a rule* $\mathtt{g}(x_1, \ldots, x_k) \to s$ *in* R *such that* $\ell_t(v) = \mathtt{g}$*,* $t/vi \in T_\Sigma$ *for every* $i \in [k]$*, and* $t' = t[v \leftarrow s]$*. The* tree language IO-generated by G *is* $L_{\mathrm{IO}}(G) = \{t \in \mathrm{T}_\Sigma \mid \mathtt{g}_{\mathrm{in}} \Rightarrow^*_{G,\mathrm{IO}} t\}$.

Example 1. We consider a very simple example of a CFTG $G_1 = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$. It has signatures $\Sigma = \{\mathtt{f}^{(2)}, \mathtt{d}^{(1)}, \mathtt{a}^{(0)}, \mathtt{c}^{(0)}\}$ and $\Xi = \{\mathtt{g}_{\mathrm{in}}^{(0)}, \mathtt{g}_1^{(2)}, \mathtt{g}_2^{(1)}, \mathtt{g}_3^{(2)}\}$, and R consists of the rules

$$\begin{array}{rcl} \mathtt{g}_{\mathrm{in}} & \to & \mathtt{g}_2(\mathtt{f}(\mathtt{a}, \mathtt{g}_1(\mathtt{a}, \mathtt{c}))), \\ \mathtt{g}_1(x_1, x_2) & \to & \mathtt{f}(x_1, x_1), \\ \mathtt{g}_2(x_1) & \to & \mathtt{g}_3(x_1, \mathtt{d}(x_1)), \text{ and} \\ \mathtt{g}_3(x_1, x_2) & \to & x_1. \end{array}$$

This grammar has exactly one derivation, viz., $\mathtt{g}_{\mathrm{in}} \Rightarrow_{G_1,\mathrm{IO}} \mathtt{g}_2(\mathtt{f}(\mathtt{a}, \mathtt{g}_1(\mathtt{a}, \mathtt{c}))) \Rightarrow_{G_1,\mathrm{IO}} \mathtt{g}_2(\mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a}))) \Rightarrow_{G_1,\mathrm{IO}} \mathtt{g}_3(\mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a})), \mathtt{d}(\mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a})))) \Rightarrow_{G_1,\mathrm{IO}} \mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a}))$, and so $L_{\mathrm{IO}}(G_1) = \{\mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a}))\}$. □

3 Jungles and their Substitution

In this section, we recall some notions regarding jungles [1, 15, 16, 18], and present some elementary properties of jungles.

Jungles can either be defined as node-labeled graphs (see, e.g., [1, 2]) or, equivalently, as edge-labeled hypergraphs (see, e.g., [5, 15, 18]). Here we choose to define them as node-labeled graphs, which are technically more convenient for our purposes (and which are closer to trees).

3.1 Jungles

Intuitively, a jungle is a directed ordered acyclic graph representing a tree. In such a jungle, subtrees can be shared and unreachable subtrees, so-called garbage, may occur.

Let Σ be a signature. A *directed ordered graph* (abbreviated DOG) over Σ is a triple $G = (V, lab, arg)$ consisting of a finite set V of *nodes*, a *labelling function* $lab\colon V \to \Sigma$, and an *argument function* $arg\colon V \to V^*$ such that $|arg(v)| = rk(lab(v))$ for every $v \in V$.

We define the *rank* of a node v as the rank of its label, i.e., $rk(v) = rk(lab(v))$. The elements of the sequence $arg(v)$ are called the *arguments* of v. In particular, the i-th element of the sequence will be denoted $arg(v, i)$, and is called the i-th argument of v. The DOG G can be visualized as an ordinary directed graph (V, E) with labelled nodes and edges, where the set of edges is $E = \{(v, arg(v, i)) \mid v \in V, i \in [rk(v)]\}$ and the label of the edge $(v, arg(v, i))$ is the natural number i. Accordingly, for nodes v and w of G, a (directed) *path from v to w* is a sequence $v_1 \cdots v_n \in V^*$ such that $n \geq 1$, $v = v_1$, $w = v_n$ and v_{j+1} is an argument of v_j for every $j \in [n-1]$. The DOG G is *acyclic* (in short, a DOAG) if, for every $v \in V$, the only path from v to v is v. A *topological order* of a DOG G is a linear order $<$ on its set V of nodes such that $arg(v, i) < v$ for every $v \in V$ and $i \in [rk(v)]$. It is well known (and easy to see) that a DOG is a DOAG if and only if it has a topological order.

A *jungle* over Σ is a DOAG with a designated node, i.e., it is a four-tuple $J = (V, res, lab, arg)$ where (V, lab, arg) is a DOAG over Σ, and $res \in V$ is the *result node* of J. The set of jungles over Σ is denoted J_Σ. A subset of J_Σ is a *jungle language*. For $k \in \mathbb{N}$, we denote $\mathrm{J}_{\Sigma \cup X_k}$ by $\mathrm{J}_\Sigma(X_k)$; it is the set of *jungles with k variables*. Note that $\mathrm{J}_\Sigma(X_0) = \mathrm{J}_\Sigma$. If necessary, the components of a jungle J will be denoted by V_J, res_J, lab_J, arg_J, respectively, and similarly for derived notions such as E_J, rk_J, etc. Two jungles J and K are *disjoint* if $V_J \cap V_K = \emptyset$. As usual, we do not distinguish between isomorphic jungles, i.e., jungles that are identical up to a bijective renaming of their nodes.

Figure 1 shows three example jungles: $K, K'' \in \mathrm{J}_\Sigma$ and $K' \in \mathrm{J}_\Sigma(X_1)$ where Σ is the signature $\{\mathtt{f}^{(2)}, \mathtt{h}^{(1)}, \mathtt{d}^{(1)}, \mathtt{a}^{(0)}, \mathtt{c}^{(0)}\}$. All edges are assumed to be directed downwards. Outgoing edges of the same node are assumed to be ordered from left to right. Result nodes are indicated by dashed circles. Thus, $K'' = (V, res, lab, arg)$ with, e.g., $V = \{d, f_1, a_1, f_2, a_2, c\}$ and $res = f_1$, $lab(d) = \mathtt{d}$, $lab(f_1) = lab(f_2) = \mathtt{f}$, $lab(a_1) = lab(a_2) = \mathtt{a}$, $lab(c) = \mathtt{c}$, $arg(d) = f_1$, $arg(f_1) = a_1 f_2$, $arg(f_2) = a_2 a_2$, and $arg(c) = \lambda$. A topological order of K'' is $c < a_2 < f_2 < a_1 < f_1 < d$.

Since trees over Σ are identified with graphs in the usual way, we will view T_Σ as a subset of J_Σ. To be precise, every tree $t \in \mathrm{T}_\Sigma$ will be identified with the jungle (V, res, lab, arg) where $V = V(t)$, $res = \lambda$, and for every $v \in V$, $lab(v) = \ell_t(v)$ and $arg(v, i) = vi$ for $i \in [rk(\ell_t(v))]$. In this way, $\mathrm{T}_\Sigma \subseteq \mathrm{J}_\Sigma$ and $\mathrm{T}_\Sigma(X_k) \subseteq \mathrm{J}_\Sigma(X_k)$ for every $k \in \mathbb{N}$.

Jungles generalize trees by allowing nodes (and hence whole subtrees) to be shared. A node w of a jungle J is shared if there are distinct pairs $(v, j), (v', j')$

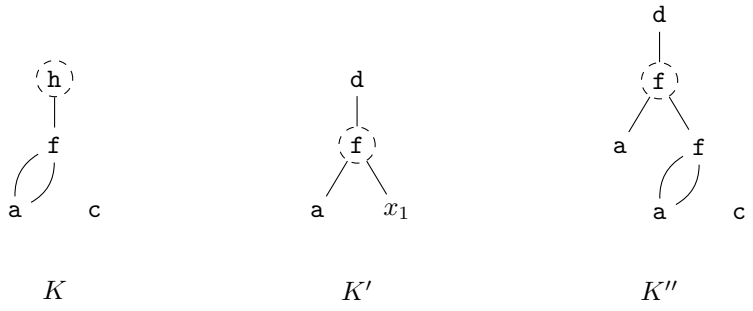

Figure 1: Jungles K, K' and K''. Using jungle substitution as defined later, $K'' = K[v \leftarrow K']$, where v is the node of K with label `h`.

such that $arg_J(v, j) = w = arg_J(v', j')$. Moreover, jungles contain garbage nodes, i.e., nodes w for which there is no path in J from res_J to w. When jungles are interpreted in a nondeterministic algebra (as we will do in Section 5), a node may have no value, one value, or several possible values. Then shared nodes and garbage nodes force a call by value semantics: a node must be evaluated, even if it will not be used (i.e., is garbage); and when it is used several times (i.e., is shared), the same value must be taken each time. In Figure 1, the node with label `a` of K is shared (and so is the node a_2 of K''), and the nodes with label `c` and `d` are garbage.

3.2 Jungle Substitution

We first show how to contract certain edges of jungles. Let `id` be a symbol of rank 1 that does not occur in Σ; intuitively, it stands for the identity function. For a jungle $J \in \mathrm{J}_{\Sigma\cup\{\mathtt{id}\}}(X_n)$, $n \in \mathbb{N}$, and a set W of nodes of J with label `id` we define $ctr_W(J) \in \mathrm{J}_{\Sigma\cup\{\mathtt{id}\}}(X_n)$ to be the result of contracting all edges $(v, arg(v, 1))$ of J with $v \in W$; the node that results from the identification of v and $arg(v, 1)$ receives the label of $arg(v, 1)$. Formally, for $J = (V, res, lab, arg)$ and $W \subseteq \{v \in V \mid lab(v) = \mathtt{id}\}$, we define the function $\gamma : V \to V \setminus W$ such that $\gamma(v) = v$ if $v \in V \setminus W$, and $\gamma(v) = \gamma(v')$ if $v \in W$ and v' is the (unique) argument of v; note that γ is well defined because J is acyclic. Then $ctr_W(J) = (V \setminus W, res', lab', arg')$ where $res' = \gamma(res)$, lab' is the restriction of lab to $V \setminus W$, and for $v \in V \setminus W$, if $arg(v) = v_1 \cdots v_k$ with $v_i \in V$ then $arg'(v) = \gamma(v_1) \cdots \gamma(v_k)$. Note that $ctr_W(J)$ is indeed acyclic: the restriction of a topological order of J to $V \setminus W$ is a topological order of $ctr_W(J)$. In particular, we define the *contraction* $ctr(J) \in \mathrm{J}_\Sigma(X_n)$ of J by $ctr(J) = ctr_W(J)$ where $W = \{v \in V \mid lab(v) = \mathtt{id}\}$. Thus, $V_{ctr(J)}$ consists of all nodes of J that do not have label `id`; the above function $\gamma\colon V_J \to V_{ctr(J)}$ is called the *track* function of J.

For a jungle J and a node v of rank k, we now show how to substitute a jungle K with k variables for that node v in J. Intuitively, the node v is replaced by the

result node of K, and every node of K with label x_i is replaced by the i-th argument of v. Note that in the special case where the result node of K has label x_i, the node v is replaced by its i-th argument.

Formally, let $J \in \mathrm{J}_\Sigma(X_n)$, $v \in V_J$ with $rk_J(v) = k$, and $K \in \mathrm{J}_\Sigma(X_k)$. We assume that J and K are disjoint, otherwise we consider a disjoint isomorphic copy of K. We first define the jungle $J\langle v \leftarrow K\rangle \in \mathrm{J}_{\Sigma\cup\{\mathtt{id}\}}(X_n)$ to be the union of J and K, with result node res_J, and with the following changes: $lab(v)$ is changed into $\mathtt{id}$ and $arg(v)$ into res_K, and for every $w \in V_K$ and $i \in [k]$, if $lab(w) = x_i$, then $lab(w)$ is changed into $\mathtt{id}$ and $arg(w)$ into $arg(v,i)$.[2] It should be clear that $U = J\langle v \leftarrow K\rangle$ is acyclic: if $<_J$ and $<_K$ are topological orders for J and K, respectively, then a topological order for U is obtained by inserting $<_K$ just before v in $<_J$, i.e., $<_U$ is the union of $<_J$, $<_K$, $\{(v', w) \mid v' \in V_J,\, v' < v,\, w \in V_K\}$ and $\{(w, v') \mid w \in V_K,\, v' \in V_J,\, v \leq v'\}$.

Finally, we define $J[v \leftarrow K] \in \mathrm{J}_\Sigma(X_n)$ to be the jungle $ctr(J\langle v \leftarrow K\rangle)$. It is called the *substitution* of K for v in J. Note that $V_{J[v\leftarrow K]}$ is the union of $V_J \setminus \{v\}$ and $V_K \setminus \{w \in V_K \mid lab_K(w) \in X\}$. Note also that the track function γ of $J\langle v \leftarrow K\rangle$ is the identity on $V_{J[v\leftarrow K]}$; moreover, $\gamma(v) = res_K$ if $lab_K(res_K) \notin X$, $\gamma(v) = \arg_J(v,i)$ if $lab_K(res_K) = x_i$, and $\gamma(w) = \arg_J(v,i)$ for every $w \in V_K$ with $lab_K(w) = x_i$. A very simple example of substitution is shown in Figure 1.

In the next section we will need the fact that jungle substitution is confluent and associative, as defined in [3]. These are natural properties that are satisfied by many notions of substitution that are used in context-free grammars for several types of structures, as shown in [3].[3] We start with a simple lemma.

Lemma 1. *For a jungle $J \in \mathrm{J}_{\Sigma\cup\{\mathtt{id}\}}(X_n)$, a node $v \in V_J$ of rank k with $lab_J(v) \neq \mathtt{id}$, and a jungle $K \in \mathrm{J}_{\Sigma\cup\{\mathtt{id}\}}(X_k)$,*

$$ctr(J)[v \leftarrow ctr(K)] = ctr(J\langle v \leftarrow K\rangle).$$

Proof. The straightforward proofs of the following two equalities are left to the reader. Let W, W_1, W_2 be sets of nodes with label $\mathtt{id}$, such that $W, W_1 \subseteq V_J$ and $W_2 \subseteq V_K$.

(i) $ctr(ctr_W(J)) = ctr(J)$

(ii) $ctr_{W_1}(J)\langle v \leftarrow ctr_{W_2}(K)\rangle = ctr_{W_1\cup W_2}(J\langle v \leftarrow K\rangle)$

By (ii), $ctr(J)[v \leftarrow ctr(K)] = ctr(ctr_{W_1\cup W_2}(J\langle v \leftarrow K\rangle))$ where $W_1 = \{w \in V_J \mid lab_J(w) = \mathtt{id}\}$ and $W_2 = \{w \in V_K \mid lab_K(w) = \mathtt{id}\}$. This equals $ctr(J\langle v \leftarrow K\rangle)$ by (i), applied to $J\langle v \leftarrow K\rangle$. □

[2] To be completely formal, $U = J\langle v \leftarrow K\rangle$ is defined as follows: $V_U = V_J \cup V_K$, $res_U = res_J$,

- $lab_U(u) = lab_J(u)$ and $arg_U(u) = arg_J(u)$ if $u \in V_J$ and $u \neq v$,
- $lab_U(v) = \mathtt{id}$ and $arg_U(v) = res_K$,
- $lab_U(u) = lab_K(u)$ and $arg_U(u) = arg_K(u)$ if $u \in V_K$ and $lab_K(u) \notin X_k$, and
- $lab_U(u) = \mathtt{id}$ and $arg_U(u) = arg_J(v,i)$ if $u \in V_K$ and $lab_K(u) = x_i$, for every $i \in [k]$.

[3] When jungles are defined as hypergraphs, jungle substitution is modeled by hyperedge replacement (see [5, Section 4]). It is well known that the corresponding notion of hypergraph substitution is confluent and associative (see, e.g., [6, Section 2.2.2]).

In the next two lemmas we show that jungle substitution is confluent and associative, respectively.

Lemma 2. *For jungles* $J \in \mathrm{J}_\Sigma(X_n)$, $K_1 \in \mathrm{J}_\Sigma(X_{k_1})$, $K_2 \in \mathrm{J}_\Sigma(X_{k_2})$ *and distinct nodes* $v_1, v_2 \in V_J$ *of rank* k_1, k_2, *respectively,*

$$J[v_1 \leftarrow K_1][v_2 \leftarrow K_2] = J[v_2 \leftarrow K_2][v_1 \leftarrow K_1].$$

Proof. By Lemma 1, $J[v_1 \leftarrow K_1][v_2 \leftarrow K_2] = ctr(J\langle v_1 \leftarrow K_1\rangle\langle v_2 \leftarrow K_2\rangle)$. It is obvious that $J\langle v_1 \leftarrow K_1\rangle\langle v_2 \leftarrow K_2\rangle = J\langle v_2 \leftarrow K_2\rangle\langle v_1 \leftarrow K_1\rangle$. □

Lemma 3. *For jungles* $J \in \mathrm{J}_\Sigma(X_n)$, $K_1 \in \mathrm{J}_\Sigma(X_{k_1})$, $K_2 \in \mathrm{J}_\Sigma(X_{k_2})$ *and nodes* $v_1 \in V_J$ *of rank* k_1 *and* $v_2 \in V_{K_1}$ *of rank* k_2 *with* $lab_{K_1}(v_2) \notin X_{k_1}$,

$$J[v_1 \leftarrow K_1][v_2 \leftarrow K_2] = J[v_1 \leftarrow K_1[v_2 \leftarrow K_2]].$$

Proof. The proof is similar to the previous one. Lemma 1 implies that both $J[v_1 \leftarrow K_1][v_2 \leftarrow K_2] = ctr(J\langle v_1 \leftarrow K_1\rangle\langle v_2 \leftarrow K_2\rangle)$ and $J[v_1 \leftarrow K_1[v_2 \leftarrow K_2]] = ctr(J\langle v_1 \leftarrow K_1\langle v_2 \leftarrow K_2\rangle\rangle)$. And it is obvious that $J\langle v_1 \leftarrow K_1\rangle\langle v_2 \leftarrow K_2\rangle = J\langle v_1 \leftarrow K_1\langle v_2 \leftarrow K_2\rangle\rangle$. □

4 Context-free Jungle Grammars

Having defined jungles and their substitution, we now define the notion of a context-free jungle grammar in an obvious way, see [5, Definition 7.6].

Definition 2. *A* context-free jungle grammar *(abbreviated CFJG) is a four-tuple* $G = (\Xi, \Sigma, R, \mathsf{g}_{\mathrm{in}})$ *such that*

- Ξ *and* Σ *are disjoint signatures of* nonterminals *and* terminals, *respectively,*
- R *is a finite set of* rules *of the form* $\mathsf{g}(x_1, \ldots, x_k) \to K$, *where* $k \in \mathbb{N}$, $\mathsf{g}^{(k)} \in \Xi$ *and* $K \in \mathrm{J}_{\Xi\cup\Sigma}(X_k)$, *and*
- $\mathsf{g}_{\mathrm{in}} \in \Xi$ *is the* initial nonterminal, *of rank* 0.

For jungles $J, J' \in \mathrm{J}_{\Xi\cup\Sigma}$, *there is a derivation step* $J \Rightarrow_G J'$ *if there are a node* $v \in V_J$ *and a rule* $\mathsf{g}(x_1, \ldots, x_k) \to K$ *in* R *such that* $lab_J(v) = \mathsf{g}$ *and* $J' = J[v \leftarrow K]$. *The* jungle language generated by G *is* $L_{\mathrm{J}}(G) = \{J \in \mathrm{J}_\Sigma \mid \mathsf{g}_{\mathrm{in}} \Rightarrow_G^* J\}$.[4]

Since every tree is a jungle, every context-free tree grammar G is also a context-free jungle grammar, generating not only the tree language $L_{\mathrm{IO}}(G)$ but also the jungle language $L_{\mathrm{J}}(G)$.

[4] Note that g_{in} is a (one-node) jungle, because g_{in} has rank 0 and $T_{\Xi\cup\Sigma} \subseteq \mathrm{J}_{\Xi\cup\Sigma}$.

Example 2. We consider a very simple example of a CFJG $G_2 = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$. It has the same terminal signature $\Sigma = \{\mathtt{f}^{(2)}, \mathtt{d}^{(1)}, \mathtt{a}^{(0)}, \mathtt{c}^{(0)}\}$ as the CFTG G_1 of Example 1. The nonterminal signature is $\Xi = \{\mathtt{g}^{(0)}, \mathtt{h}^{(1)}\}$ with $\mathtt{g}_{\mathrm{in}} = \mathtt{g}$, and the set R consists of the two rules $\mathtt{g} \to K$ and $\mathtt{h}(x_1) \to K'$, where K and K' are given in Figure 1. The unique derivation of this grammar is $\mathtt{g} \Rightarrow_G K \Rightarrow_G K[v \leftarrow K']$, where v is the node of K with label $\mathtt{h}$. Thus $L_{\mathrm{J}}(G_2) = \{K''\}$, where K'' is given in Figure 1.

As another simple example we consider the context-free jungle grammar G_1 of Example 1. A derivation of G_1 is shown in Figure 2; it generates the jungle K''. The other two derivations of G_1 also generate the jungle K'' (in accordance with Lemma 2). Thus, $L_{\mathrm{J}}(G_1) = L_{\mathrm{J}}(G_2) = \{K''\}$.

An interpretation of grammars G_2 and G_1 will be given in Examples 5 and 6. □

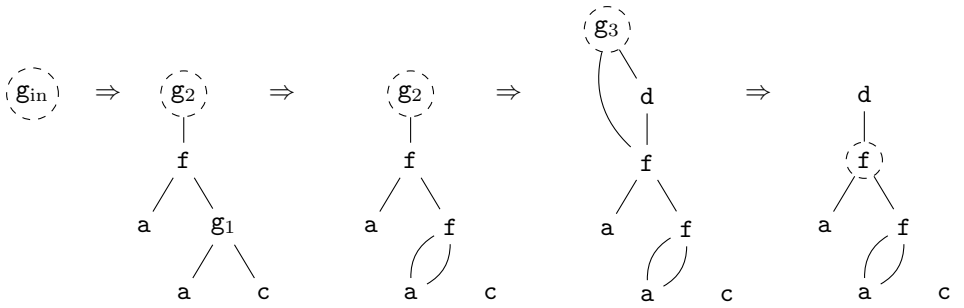

Figure 2: A derivation of G_1.

Our next aim is to show that rules of a CFJG can be substituted into each other, without changing the generated jungle language. More precisely, consider a rule $\mathtt{g}(x_1, \ldots, x_k) \to K$ and let $v \in V_K$ be a node of rank m with nonterminal label $\mathtt{h}$. Then this rule can be replaced by all rules

$$\mathtt{g}(x_1, \ldots, x_k) \to K[v \leftarrow K']$$

where K' is the right-hand side of a rule with left-hand side $\mathtt{h}(x_1, \ldots, x_m)$. This clearly holds for the CFJG G_2 of Example 2: the resulting grammar has rules $\mathtt{g} \to K''$ and $\mathtt{h}(x_1) \to K'$, and thus generates the same jungle language $\{K''\}$; note that the second rule has become superfluous.

The above property is well known for context-free string grammars, and for several types of context-free graph grammars. Its proof is based on the fact that jungle substitution is confluent and associative, as shown in the previous section.

Lemma 4. *Let $G = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$ be a CFJG. Let $K \in \mathrm{J}_{\Xi \cup \Sigma}$ and let $v \in V_K$ be such that $lab_K(v) = \mathtt{h}^{(m)} \in \Xi$. Then, for every $J \in \mathrm{J}_\Sigma$ and $n \in \mathbb{N}$, $K \Rightarrow_G^n J$ if and only if there exists a rule $\mathtt{h}(x_1, \ldots, x_m) \to K'$ in R such that $K[v \leftarrow K'] \Rightarrow_G^{n-1} J$.*

Proof. The if direction is obvious, because $K \Rightarrow_G K[v \leftarrow K']$. The only-if direction is proved by induction on n. Let $K \Rightarrow_G K[w \leftarrow L] \Rightarrow_G^{n-1} J$ be the first step of the derivation. If $w = v$ then there is a rule $\mathtt{h}(x_1, \ldots, x_m) \to L$ in R, and we are ready. Now assume that $w \neq v$. By the induction hypothesis, there is a rule $\mathtt{h}(x_1, \ldots, x_m) \to K'$ in R such that $K[w \leftarrow L][v \leftarrow K'] \Rightarrow_G^{n-2} J$. Hence $K[v \leftarrow K'][w \leftarrow L] \Rightarrow_G^{n-2} J$ by Lemma 2. This implies that

$$K[v \leftarrow K'] \Rightarrow_G K[v \leftarrow K'][w \leftarrow L] \Rightarrow_G^{n-2} J,$$

and so $K[v \leftarrow K'] \Rightarrow_G^{n-1} J$. □

Theorem 1. *Let $G = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$ be a CFJG. Let $\mathtt{g}(x_1, \ldots, x_k) \to K$ be a rule in R and let $v \in V_K$ be such that $lab_K(v) = \mathtt{h}^{(m)} \in \Xi$. Let G' be the CFJG $(\Xi, \Sigma, R', \mathtt{g}_{\mathrm{in}})$ where R' is obtained from R by replacing the rule $\mathtt{g}(x_1, \ldots, x_k) \to K$ by all rules $\mathtt{g}(x_1, \ldots, x_k) \to K[v \leftarrow K']$ where $\mathtt{h}(x_1, \ldots, x_m) \to K'$ is in R. Then $L_{\mathrm{J}}(G') = L_{\mathrm{J}}(G)$.*

Proof. We prove by induction on the length of the derivations that for all $I \in \mathrm{J}_{\Xi \cup \Sigma}$ and $J \in \mathrm{J}_\Sigma$, $I \Rightarrow_G^* J$ if and only if $I \Rightarrow_{G'}^* J$.

For the only-if direction, we consider the first step of the derivation $I \Rightarrow_G^* J$. It clearly suffices to consider the case that the rule $\mathtt{g}(x_1, \ldots, x_k) \to K$ is applied in this step. Thus, let $I \Rightarrow_G I[w \leftarrow K] \Rightarrow_G^* J$, where $lab_I(w) = \mathtt{g}$. By Lemma 4 there is a rule $\mathtt{h}(x_1, \ldots, x_m) \to K'$ in R such that $I[w \leftarrow K][v \leftarrow K'] \Rightarrow_G^* J$, and this derivation is shorter than the derivation $I \Rightarrow_G^* J$. Hence, by the induction hypothesis, $I[w \leftarrow K][v \leftarrow K'] \Rightarrow_{G'}^* J$. Now, by Lemma 3, we have $I[w \leftarrow K][v \leftarrow K'] = I[w \leftarrow K[v \leftarrow K']]$, and so $I \Rightarrow_{G'} I[w \leftarrow K[v \leftarrow K']] \Rightarrow_{G'}^* J$, where the rule $\mathtt{g}(x_1, \ldots, x_k) \to K[v \leftarrow K']$ of G' is applied in the first step.

The if direction is similar, but slightly easier. For the first step of the derivation $I \Rightarrow_{G'}^* J$ it suffices to consider a rule $\mathtt{g}(x_1, \ldots, x_k) \to K[v \leftarrow K']$ in R'. If $I \Rightarrow_{G'} I[w \leftarrow K[v \leftarrow K']] \Rightarrow_{G'}^* J$, then $I[w \leftarrow K][v \leftarrow K'] \Rightarrow_G^* J$ by Lemma 3 and the induction hypothesis, and so $I \Rightarrow_G I[w \leftarrow K] \Rightarrow_G^* J$ by Lemma 4. □

Before proving our main result, we discuss an easy normal form of context-free jungle grammars. A CFJG $G = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$ is in *variable normal form* if, for every rule $\mathtt{g}(x_1, \ldots, x_k) \to K$ in R and every $i \in [k]$, exactly one node of K has label x_i. It is easy to see that for every CFJG G an equivalent CFJG G' can be constructed that is in variable normal form, as follows. If $\mathtt{g}(x_1, \ldots, x_k) \to K$ is a rule of G, then $\mathtt{g}(x_1, \ldots, x_k) \to K'$ is a rule of G', where K' is obtained from K by identifying all nodes with label x_i, for each $i \in [k]$, and adding an isolated node with label x_i if K does not have such a node. To be precise, $K' = \mathtt{g}(x_1, \ldots, x_k)[\lambda \leftarrow K]$, i.e., the substitution of K for the node with label $\mathtt{g}$ in the jungle $\mathtt{g}(x_1, \ldots, x_k)$. It follows from Lemma 3 (and is also easy to see) that $J[v \leftarrow K'] = J[v \leftarrow K]$ for every jungle J and every node $v \in V_J$ with label $\mathtt{g}$, and hence $L_{\mathrm{J}}(G') = L_{\mathrm{J}}(G)$.

The equivalence of G' and G can also be proved by Theorem 1, as follows. Let G_1 be the CFJG obtained from G by replacing every rule $\mathtt{g}(x_1, \ldots, x_k) \to K$ by the rules $\mathtt{g}(x_1, \ldots, x_k) \to \mathtt{g}_0(x_1, \ldots, x_k)$ and $\mathtt{g}_0(x_1, \ldots, x_k) \to K$, where $\mathtt{g}_0$ is a

new nonterminal. Obviously, $L_{\mathrm{J}}(G_1) = L_{\mathrm{J}}(G)$. Application of Theorem 1 to each rule $\mathtt{g}(x_1, \ldots, x_k) \to \mathtt{g}_0(x_1, \ldots, x_k)$, changes G_1 into the equivalent grammar G_1' in which that rule is changed into all rules $\mathtt{g}(x_1, \ldots, x_k) \to K'$. Since the rules $\mathtt{g}_0(x_1, \ldots, x_k) \to K$ have become useless in G_1', we obtain that $L_{\mathrm{J}}(G_1') = L_{\mathrm{J}}(G')$ and hence $L_{\mathrm{J}}(G) = L_{\mathrm{J}}(G')$.

We now show the main result of this paper: context-free tree grammars have the same jungle generating power as context-free jungle grammars.

Theorem 2. *For every context-free jungle grammar G there is a context-free tree grammar H such that $L_{\mathrm{J}}(H) = L_{\mathrm{J}}(G)$.*

Proof. Let $G = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$ be a CFJG in variable normal form, which can be assumed by the discussion above. Consider a rule $\mathtt{g}(x_1, \ldots, x_k) \to K$ in R. We will construct an equivalent CFJG G' in which this rule is replaced by a finite set of CFTG rules. By repeating this construction we obtain the required CFTG H. In the CFJG $G' = (\Xi', \Sigma, R', \mathtt{g}_{\mathrm{in}})$, the rule will be simulated by a sequence of rules that build the jungle K node by node, in a bottom-up fashion. This is similar to, but slightly more complicated than, the construction of Chomsky normal form for context-free string grammars.

Let $v_1 < \cdots < v_k < v_{k+1} < \cdots < v_{k+\ell}$, $\ell \geq 0$, be a topological order of K such that $v_1, \ldots, v_k$ are the (unique) nodes of K with labels $x_1, \ldots, x_k$, respectively. Obviously such a topological order exists, because the nodes $v_1, \ldots, v_k$ have no arguments. We define $\Xi' = \Xi \cup \{\mathtt{g}_0, \mathtt{g}_1, \ldots, \mathtt{g}_\ell\}$ where $\mathtt{g}_i$ is a new nonterminal of rank $k + i$, for $0 \leq i \leq \ell$. Moreover, we define R' to be the set of rules obtained from R by replacing the rule $\mathtt{g}(x_1, \ldots, x_k) \to K$ by the CFTG rules

- $\mathtt{g}(x_1, \ldots, x_k) \to \mathtt{g}_0(x_1, \ldots, x_k)$,
- $\mathtt{g}_{i-1}(x_1, \ldots, x_{k+i-1}) \to \mathtt{g}_i(x_1, \ldots, x_{k+i-1}, \mathtt{f}_i(x_{j_1}, \ldots, x_{j_p}))$ for $i \in [\ell]$, where $\mathtt{f}_i^{(p)} = \mathit{lab}_K(v_{k+i})$ and $\mathit{arg}_K(v_{k+i}, q) = v_{j_q}$ for every $q \in [p]$, and
- $\mathtt{g}_\ell(x_1, \ldots, x_{k+\ell}) \to x_j$, where $\mathit{res}_K = v_j$.

For $i \in [\ell + 1]$, let t_i be the right-hand side of the rule with left-hand side $\mathtt{g}_{i-1}(x_1, \ldots, x_{k+i-1})$. Note that in the second item $j_q \in [k + i - 1]$, because $<$ is a topological order. Thus, $t_i \in \mathrm{T}_{\Xi \cup \Sigma}(X_{k+i-1})$ as required.

To prove the correctness of this construction, we take new nodes $r_0, \ldots, r_\ell$, we assume that the nodes in the right-hand side $\mathtt{g}_0(x_1, \ldots, x_k)$ of the first rule are $r_0, v_1, \ldots, v_k$ with respective labels $\mathtt{g}_0, x_1, \ldots, x_k$, and we assume for $i \in [\ell]$, that the nodes in t_i with labels $\mathtt{g}_i$ and $\mathtt{f}_i$ are r_i and v_{k+i}, respectively. For $0 \leq i \leq \ell$, we define the jungle $K_i = (V, \mathit{res}, \mathit{lab}, \mathit{arg}) \in \mathrm{J}_{\Xi' \cup \Sigma}(X_k)$ as follows: $V = \{r_i, v_1, \ldots, v_{k+i}\}$, $\mathit{res} = r_i$, $\mathit{lab}(r_i) = \mathtt{g}_i$, $\mathit{arg}(r_i) = v_1 \cdots v_{k+i}$, and for $j \in [k + i]$, $\mathit{lab}(v_j) = \mathit{lab}_K(v_j)$ and $\mathit{arg}(v_j) = \mathit{arg}_K(v_j)$. Moreover, we define $K_{\ell+1} = K$.

It can easily be checked that $K_0 = \mathtt{g}_0(x_1, \ldots, x_k)$ and $K_i = K_{i-1}[r_{i-1} \leftarrow t_i]$ for every $i \in [\ell + 1]$. Thus, by iterated application of Theorem 1 (i.e., formally by induction on i), the grammar G' is equivalent to the grammar G_i' that is obtained from G' by changing the rule $\mathtt{g}(x_1, \ldots, x_k) \to \mathtt{g}_0(x_1, \ldots, x_k)$ into the rule

$\mathtt{g}(x_1, \ldots, x_k) \to K_i$. Hence, since $K_{\ell+1} = K$, the grammar $G'_{\ell+1}$ has all rules of G plus all rules $\mathtt{g}_i(x_1, \ldots, x_{k+i}) \to t_{i+1}$ for $0 \leq i \leq \ell$. However, since $\mathtt{g}_0$ does not appear in any right-hand side of a rule of $G'_{\ell+1}$, the rules $\mathtt{g}_i(x_1, \ldots, x_{k+i}) \to t_{i+1}$ have become useless in $G'_{\ell+1}$, and hence $G'_{\ell+1}$ is equivalent to G.

This shows that $L_{\mathrm{J}}(G') = L_{\mathrm{J}}(G)$. □

Example 3. We illustrate the construction in the proof of Theorem 2 with the CFJG G_2 of Example 2. It has initial nonterminal $\mathtt{g}$ and it has the two rules $\mathtt{g} \to K$ and $\mathtt{h}(x_1) \to K'$, where K and K' are given in Figure 1. The resulting CFTG H has the same initial nonterminal $\mathtt{g}$. Its rules are constructed based on the following topological orders of K and K' (where we indicate nodes by their labels): $\mathtt{a} < \mathtt{c} < \mathtt{f} < \mathtt{h}$ for K, and $x_1 < \mathtt{a} < \mathtt{f} < \mathtt{d}$ for K'. This gives the following rules:

$$\begin{aligned}
\mathtt{g} &\to \mathtt{g}_0 \\
\mathtt{g}_0 &\to \mathtt{g}_1(\mathtt{a}) \\
\mathtt{g}_1(x_1) &\to \mathtt{g}_2(x_1, \mathtt{c}) \\
\mathtt{g}_2(x_1, x_2) &\to \mathtt{g}_3(x_1, x_2, \mathtt{f}(x_1, x_1)) \\
\mathtt{g}_3(x_1, x_2, x_3) &\to \mathtt{g}_4(x_1, x_2, x_3, \mathtt{h}(x_3)) \\
\mathtt{g}_4(x_1, x_2, x_3, x_4) &\to x_4 \\
\mathtt{h}(x_1) &\to \mathtt{h}_0(x_1) \\
\mathtt{h}_0(x_1) &\to \mathtt{h}_1(x_1, \mathtt{a}) \\
\mathtt{h}_1(x_1, x_2) &\to \mathtt{h}_2(x_1, x_2, \mathtt{f}(x_2, x_1)) \\
\mathtt{h}_2(x_1, x_2, x_3) &\to \mathtt{h}_3(x_1, x_2, x_3, \mathtt{d}(x_3)) \\
\mathtt{h}_3(x_1, x_2, x_3, x_4) &\to x_3
\end{aligned}$$

Of course, this is not the simplest CFTG that generates the same jungle language as G_2. A simpler one is the grammar G_1 of Example 1, as we saw in Example 2. □

Thus, context-free tree grammars have the same jungle generating power as context-free jungle grammars. Vice versa, every jungle represents a tree (by unfolding) and we will show that context-free jungle grammars have the same tree generating power as context-free tree grammars.

Every jungle $J \in \mathrm{J}_\Sigma$ represents a unique tree $\mathrm{tree}(J) \in \mathrm{T}_\Sigma$, namely $\mathrm{tree}(J) = \mathrm{tree}(J, \mathit{res}_J)$, where $\mathrm{tree}(J, v)$ is defined as follows, for all $v \in V_J$: if $\mathit{lab}_J(v) = \mathtt{f}^{(k)}$ and $\mathit{arg}_J(v) = v_1 \cdots v_k$, then $\mathrm{tree}(J, v) = \mathtt{f}(\mathrm{tree}(J, v_1), \ldots, \mathrm{tree}(J, v_k))$. For example, $\mathrm{tree}(K'') = \mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a}))$ where K'' is given in Figure 1.

For a context-free jungle grammar G we define the *tree language generated by* G as $L_{\mathrm{T}}(G) = \{\mathrm{tree}(J) \mid J \in L_{\mathrm{J}}(G)\}$.

Theorem 3. *A tree language can be generated by a context-free jungle grammar if and only if it can be IO-generated by a context-free tree grammar.*

Proof. (If) It is shown in [5, Corollary 6.5] that $L_{\mathrm{T}}(G) = L_{\mathrm{IO}}(G)$ for every context-free tree grammar G.[5]

[5] As an example, $L_{\mathrm{IO}}(G_1) = \{\mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a}))\}$ for the context-free tree grammar G_1 of Example 1, and, by Example 2, $L_{\mathrm{J}}(G_1) = \{K''\}$ and so $L_{\mathrm{T}}(G_1) = \{\mathrm{tree}(K'')\} = \{\mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a}))\}$.

(Only if) For a context-free jungle grammar G, let H be a context-free tree grammar such that $L_{\mathrm{J}}(H) = L_{\mathrm{J}}(G)$, which exists by Theorem 2. By the previous paragraph, $L_{\mathrm{IO}}(H) = L_{\mathrm{T}}(H)$ and so $L_{\mathrm{IO}}(H) = \{\mathrm{tree}(J) \mid J \in L_{\mathrm{J}}(H)\} = \{\mathrm{tree}(J) \mid J \in L_{\mathrm{J}}(G)\} = L_{\mathrm{T}}(G)$. □

Related results are proved in [7, 9, 12, 10]. It is shown in [7] (see also [8]) that IO context-free tree grammars generate the same tree languages as attribute grammars with one synthesized attribute. As shown in [9], arbitrary attribute grammars generate the same tree languages as jungle generating context-free graph grammars (which generalize context-free jungle grammars). In [12] it is proved that total deterministic macro tree transducers compute the same tree translations as top-down tree-to-jungle transducers. Finally, in [10] context-free tree grammars are considered such that for every rule $\mathtt{g}(x_1, \ldots, x_k) \to s$, each parameter x_i occurs exactly once in s; it should be clear that $L_{\mathrm{IO}}(G) = L_{\mathrm{J}}(G) = L_{\mathrm{T}}(G)$ for such a grammar G (cf. [5, Theorem 6.7]).

5 Jungle Delegation Networks

In this section we interpret context-free jungle grammars in algebras and call them jungle delegation networks.

5.1 Interpretations and Algebras

We wish to interpret every symbol of a signature Σ as an operation on the elements of a set $\mathbb{A}$, and to use this interpretation for evaluating jungles over Σ. Usually, the interpretation of a symbol $\mathtt{f}^{(k)}$ would be a k-ary function $f\colon \mathbb{A}^k \to \mathbb{A}$. However, we wish to consider the more general case of nondeterministic operations (as in [13, Section II.2]; see also [5, 11]). For this purpose, symbols are interpreted as relations $f \subseteq \mathbb{A}^k \times \mathbb{A}$ rather than as functions. Of course, functions and partial functions are special cases.

A Σ-*interpretation into* $\mathbb{A}$ is a function σ that maps every symbol $\mathtt{f}^{(k)} \in \Sigma$ to a relation $\sigma(\mathtt{f}) \subseteq \mathbb{A}^k \times \mathbb{A}$; in particular, if $k = 0$ then $\sigma(\mathtt{f}) \subseteq \mathbb{A}$. The pair $(\mathbb{A}, \sigma)$ is called a (*nondeterministic*) Σ-*algebra*. If $\sigma(\mathtt{f})$ is a function for all $\mathtt{f} \in \Sigma$, then $(\mathbb{A}, \sigma)$ is a *deterministic* Σ-*algebra*.

Jungles with n variables can now be interpreted as derived operations of a given Σ-algebra $(\mathbb{A}, \sigma)$, in an obvious way (see [5, Definition 5.2]).

Definition 3 (jungle evaluation)**.** *Consider a* Σ*-algebra* $(\mathbb{A}, \sigma)$ *and a jungle* $J \in \mathrm{J}_\Sigma(X_n)$*. Given* $a_1, \ldots, a_n \in \mathbb{A}$*, let* $ASS_{J,\sigma}(a_1, \ldots, a_n)$ *be the set of all assignments (i.e., functions)* $\alpha\colon V_J \to \mathbb{A}$ *such that every* $v \in V_J$*:*

- *if* $lab_J(v) = x_i$*, then* $\alpha(v) = a_i$*; and*
- *if* $lab_J(v) = \mathtt{f} \in \Sigma$ *and* $arg_J(v) = v_1 \cdots v_k$*, then* $\alpha(v) \in f(\alpha(v_1), \ldots, \alpha(v_k))$ *where* $f = \sigma(\mathtt{f})$*.*

Now, $\sigma(J) \subseteq \mathbb{A}^n \times \mathbb{A}$ is the relation given by

$$\sigma(J)(a_1, \ldots, a_n) = \{\alpha(res_J) \mid \alpha \in ASS_{J,\sigma}(a_1, \ldots, a_n)\},$$

for all $a_1 \ldots, a_n \in \mathbb{A}$. For a set of jungles $\mathcal{J} \subseteq \mathrm{J}_\Sigma(X_n)$, $\sigma(\mathcal{J}) = \bigcup_{J \in \mathcal{J}} \sigma(J)$.

Since every tree is a jungle, this also defines $\sigma(t)$ for every tree $t \in \mathrm{T}_\Sigma(X_n)$. It should be clear that $\sigma(t)$ is the usual evaluation of t in a (nondeterministic) Σ-algebra, see, e.g., [5, Lemma 5.3]. For a set of trees $\mathcal{T} \subseteq \mathrm{T}_\Sigma(X_n)$, $\sigma(\mathcal{T})$ is called the derived relation of $\mathcal{T}$ over $(\mathbb{A}, \sigma)$ in [11, Definition 5.8].

Example 4. We extend [5, Example 5.1]. Let $\Sigma = \{\mathtt{f}^{(2)}, \mathtt{d}^{(1)}, \mathtt{a}^{(0)}, \mathtt{c}^{(0)}\}$ be the signature of Examples 1 and 2, and consider the Σ-algebra $(\mathbb{A}, \sigma)$ where $\mathbb{A} = \{\diamondsuit, \heartsuit\}^*$, $\sigma(\mathtt{f})$ is string concatenation, $\sigma(\mathtt{a}) = \{\diamondsuit, \heartsuit\}$, $\sigma(\mathtt{c}) = \{\diamondsuit\}$, and $\sigma(\mathtt{d})$ is the partial function $d : \mathbb{A} \to \mathbb{A}$ such that, for every string $w \in \mathbb{A}$, $d(\diamondsuit w) = w$ and $d(\heartsuit w)$ and $d(\lambda)$ are undefined (thus, d checks that the first symbol of a string is diamonds, and deletes that symbol). Now consider the interpretation $\sigma(K'')$ of the jungle K'' of Figure 1. If one constructs $\alpha \in ASS_{K'',\sigma}$ in a bottom-up fashion, then $\alpha(res_{K''})$ can have the values $\diamondsuit\diamondsuit\diamondsuit$, $\diamondsuit\heartsuit\heartsuit$, $\heartsuit\diamondsuit\diamondsuit$, and $\heartsuit\heartsuit\heartsuit$ (note that the last two symbols are equal because of the shared node with label $\mathtt{a}$). Thus, due to the presence of the node with label $\mathtt{d}$, we have $\sigma(K'') = \{\diamondsuit\diamondsuit\diamondsuit, \diamondsuit\heartsuit\heartsuit\}$. If we redefine $\sigma(\mathtt{c}) = \emptyset$, then $\sigma(K'') = \emptyset$ because no value can be assigned to the node with label $\mathtt{c}$.

The jungle K' of Figure 1 is interpreted as the function $\sigma(K') = k' \colon \mathbb{A} \to \mathbb{A}$ such that $k'(w) = \diamondsuit w$ for every $w \in \mathbb{A}$. □

5.2 Delegation Networks

We are now ready to give the formal definition of delegation networks.

Definition 4. *A* jungle delegation network *is a system $\mathcal{N} = (G, \mathbb{A}, \sigma)$, where $G = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$ is a context-free jungle grammar and $(\mathbb{A}, \sigma)$ is a Σ-algebra. If G is a context-free tree grammar, then $\mathcal{N}$ is a* tree delegation network.[6]

The signature $\Xi \cup \Sigma$ is denoted by $\Sigma_\mathcal{N}$. For $\mathtt{g}^{(k)} \in \Xi$ we denote by $\mathrm{rhs}_G(\mathtt{g})$ the set of right-hand sides of rules in R with left-hand side $\mathtt{g}(x_1, \ldots, x_k)$.

The semantics of $\mathcal{N}$ is obtained by defining a $\Sigma_\mathcal{N}$-interpretation $\sigma_\mathcal{N}$ into $\mathbb{A}$ that agrees with σ on Σ. Since the rules of G are recursive, it is natural to choose a least fixed point semantics, using Kleene's fixed point theorem which we state next (see, e.g., [13, Theorem I.4.8]).

Proposition 1. *Let C be a complete lattice, and let $\varphi \colon C \to C$ be an ω-continuous function. Then φ has a least fixed point, and this least fixed point is equal to the least upper bound of all $\varphi^m(0)$, $m \in \mathbb{N}$, where 0 is the zero element of C.*

[6] A tree delegation network $\mathcal{N}$ is called a finitary delegation network in [5]. In [11, Definition 5.1] the syntactic part G of $\mathcal{N}$ is called a system of context-free Σ-equations.

We recall that a complete lattice is a set C with a partial order $\leq$ such that every subset of C has a least upper bound. Moreover, ω-continuity of φ means that if $c_0 \leq c_1 \leq c_2 \leq \cdots$ (with $c_i \in C$) and $c \in C$ is the least upper bound of $\{c_i \mid i \in \mathbb{N}\}$, then $\varphi(c)$ is the least upper bound of $\{\varphi(c_i) \mid i \in \mathbb{N}\}$. The zero element 0 of C is its smallest element (i.e., the least upper bound of $\emptyset$).

As is well known, the set of all relations $r \subseteq \mathbb{A}^k \times \mathbb{A}$ is a complete lattice with $\subseteq$ as partial order. We extend this ordering to $\Sigma_{\mathcal{N}}$-interpretations τ, τ' into $\mathbb{A}$ in the usual way: $\tau \leq \tau'$ if and only if $\tau(\mathtt{f}) \subseteq \tau'(\mathtt{f})$ for all $\mathtt{f} \in \Sigma_{\mathcal{N}}$. With the partial order $\leq$, the set of all $\Sigma_{\mathcal{N}}$-interpretations into $\mathbb{A}$ is a complete lattice, as is also well known. If T is a set of such interpretations, with least upper bound υ, then, for every $\mathtt{f} \in \Sigma_{\mathcal{N}}$, $\upsilon(\mathtt{f})$ is the union of all $\tau(\mathtt{f})$, $\tau \in T$. Note that the zero element 0 of the lattice is the $\Sigma_{\mathcal{N}}$-interpretation such that $0(\mathtt{f}) = \emptyset$ for all $\mathtt{f} \in \Sigma_{\mathcal{N}}$.

The semantics of $\mathcal{N}$ is a subset of $\mathbb{A}$. It will be called the language defined by $\mathcal{N}$, generalizing the notions of string language, tree language, graph language, picture language, etc. The semantics of $\mathcal{N}$ is an obvious generalization of the one for tree delegation networks (see [5, Definition 2.4]). Intuitively, G is viewed as a system of equations $(\mathtt{g} = \mathrm{rhs}_G)_{\mathtt{g} \in \Xi}$, where rhs_G is viewed as the union of its elements, and these equations are solved in the algebra $(\mathbb{A}, \sigma)$.

Definition 5. *Let $\mathcal{N} = (G, \mathbb{A}, \sigma)$ be a jungle delegation network such that $G = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$. Let $C_{\mathcal{N}}$ be the complete lattice of all $\Sigma_{\mathcal{N}}$-interpretations into $\mathbb{A}$.*

1. *The function $\varphi_{\mathcal{N}} \colon C_{\mathcal{N}} \to C_{\mathcal{N}}$ is defined as follows for every $\tau \in C_{\mathcal{N}}$:*
   - $\varphi_{\mathcal{N}}(\tau)(\mathtt{f}) = \sigma(\mathtt{f})$ *for every* $\mathtt{f} \in \Sigma$, *and*
   - $\varphi_{\mathcal{N}}(\tau)(\mathtt{g}) = \tau(\mathrm{rhs}_G(\mathtt{g}))$ *for every* $\mathtt{g} \in \Xi$.
2. *The least fixed point of $\varphi_{\mathcal{N}}$ is denoted by $\sigma_{\mathcal{N}}$.*[7]
3. *The* language defined by $\mathcal{N}$ *is* $L(\mathcal{N}) = \sigma_{\mathcal{N}}(\mathtt{g}_{\mathrm{in}})$.

Note that the language defined by $\mathcal{N}$ is a subset of $\mathbb{A}$, because the rank of $\mathtt{g}_{\mathrm{in}}$ is 0. Note furthermore that, for $\mathtt{f} \in \Sigma$, we have $\sigma_{\mathcal{N}}(\mathtt{f}) = \sigma(\mathtt{f})$.

As already observed in [5], if $\mathcal{N}$ is a tree delegation network, then, for every $\mathtt{g} \in \Xi$, the relation $\sigma_{\mathcal{N}}(\mathtt{g})$ is what is called the call by value relation computed by G over $(\mathbb{A}, \sigma)$ in the discussion after Corollary 5.7 in [11].

Example 5. Consider the jungle delegation network $\mathcal{N}_2 = (G_2, \mathbb{A}, \sigma)$ where G_2 is the CFJG of Example 2 and $(\mathbb{A}, \sigma)$ is the string algebra defined in Example 4. Since K' does not contain nonterminal symbols, we obtain that $\sigma_{\mathcal{N}_2}(\mathtt{h}) = \sigma(K')$, which is the function k' that puts $\diamond$ in front of a string (as observed in the latter example). This implies that $\sigma_{\mathcal{N}_2}(\mathtt{g}) = \sigma'(K)$, where σ' extends σ with $\sigma'(\mathtt{h}) = k'$. From this it is easy to see that $\mathcal{N}_2$ defines the language $L(\mathcal{N}_2) = \sigma_{\mathcal{N}_2}(\mathtt{g}) = \{\diamondsuit\diamondsuit\diamondsuit, \diamondsuit\heartsuit\heartsuit\}$.

[7] By Proposition 1, $\sigma_{\mathcal{N}}$ exists, as it is easy to verify that $\varphi_{\mathcal{N}}$ is ω-continuous. Note that by Definition 3, if $\tau_0 \leq \tau_1 \leq \tau_2 \leq \cdots$ (with $\tau_i \in C_{\mathcal{N}}$), $\tau \in C_{\mathcal{N}}$ is the least upper bound of $\{\tau_i \mid i \in \mathbb{N}\}$, and $\mathcal{J} \subseteq \mathrm{J}_\Sigma(X_n)$ is finite, then $\tau(\mathcal{J})$ is the union of all $\tau_i(\mathcal{J})$, $i \in \mathbb{N}$.

We have seen in Example 2 that $L_{\mathrm{J}}(G_2) = \{K''\}$, where K'' is given in Figure 1. Combining this with the fact that $\sigma(K'') = \{\diamondsuit\diamondsuit\diamondsuit, \diamondsuit\heartsuit\heartsuit\}$ (as seen in Example 4), we find that $L(\mathcal{N}_2) = \sigma(L_{\mathrm{J}}(G_2))$. In other words, $L(\mathcal{N}_2)$ equals the interpretation of the jungle language generated by its CFJG G_2. We will prove in Theorem 6 that this is a result that holds in general. □

To express the semantics of a substitution $J[v \leftarrow K]$ in terms of that of J and K, we need some terminology. Let v have rank k, let @ be a new symbol of rank k, and let $J[v \leftarrow @]$ be the jungle obtained from J by changing the label of v into @, i.e., $J[v \leftarrow @] = J[v \leftarrow @(x_1, \ldots, x_k)]$. The next lemma shows that the semantics of $J[v \leftarrow K]$ is equal to the semantics of $J[v \leftarrow @]$, when @ is interpreted as the semantics of K. For a Σ-algebra $(\mathbb{A}, \sigma)$ and a relation $r \subseteq \mathbb{A}^k \times \mathbb{A}$, we denote by $\sigma\langle @ := r\rangle$ the $(\Sigma \cup \{@\})$-interpretation σ' into $\mathbb{A}$ such that $\sigma'(@) = r$ and $\sigma'(\mathtt{f}) = \sigma(\mathtt{f})$ for every $\mathtt{f} \in \Sigma$.

Lemma 5. *Let $(\mathbb{A}, \sigma)$ be a Σ-algebra. Let $J \in \mathrm{J}_\Sigma(X_n)$, $v \in V_J$ with $rk_J(v) = k$ and $lab_J(v) \in \Sigma$, and $K \in \mathrm{J}_\Sigma(X_k)$. Then $\sigma(J[v \leftarrow K]) = \sigma\langle @ := \sigma(K)\rangle(J[v \leftarrow @])$.*

Proof. Without loss of generality we assume that $\mathtt{id} \in \Sigma$, that $\sigma(\mathtt{id})$ is the identity on $\mathbb{A}$, and that $\mathtt{id}$ does not occur in J and K. Obviously, $\sigma(I) = \sigma(ctr(I))$ for every $I \in \mathrm{J}_\Sigma(X_n)$. Hence $\sigma(J[v \leftarrow K]) = \sigma(J\langle v \leftarrow K\rangle)$.

If $\alpha \in ASS_{J\langle v \leftarrow K\rangle, \sigma}(a_1, \ldots, a_n)$, then the restriction α_K of α to V_K is in $ASS_{K,\sigma}(b_1, \ldots, b_k)$ where $b_i = \alpha(arg_J(v, i))$. Moreover, $\alpha_K(res_K) = \alpha(v)$. This shows that the restriction α_J of α to V_J is in $ASS_{J[v \leftarrow @],\, \sigma\langle @ := \sigma(K)\rangle}(a_1, \ldots, a_n)$, and so $\sigma(J[v \leftarrow K]) \subseteq \sigma\langle @ := \sigma(K)\rangle(J[v \leftarrow @])$.

If $\alpha_J \in ASS_{J[v \leftarrow @],\, \sigma\langle @ := \sigma(K)\rangle}(a_1, \ldots, a_n)$, then there exists an assignment $\alpha_K \in ASS_{K,\sigma}(b_1, \ldots, b_k)$ such that $b_i = \alpha_J(arg_J(v, i))$ and $\alpha_K(res_K) = \alpha_J(v)$. It is now clear that $\alpha_J \cup \alpha_K$ is in $ASS_{J\langle v \leftarrow K\rangle, \sigma}(a_1, \ldots, a_n)$, from which we can conclude that $\sigma\langle @ := \sigma(K)\rangle(J[v \leftarrow @]) \subseteq \sigma(J[v \leftarrow K])$. □

The next theorem is similar to Theorem 1 in Section 4. It shows that rules of the grammar of a jungle delegation network can be substituted into each other, without changing the language defined by the network.

Theorem 4. *Let $\mathcal{N} = (G, \mathbb{A}, \sigma)$ and $\mathcal{N}' = (G', \mathbb{A}, \sigma)$ be jungle delegation networks, where G and G' are as in Theorem 1. Then $L(\mathcal{N}') = L(\mathcal{N})$.*

Proof. By assumption, $G = (\Xi, \Sigma, R, \mathtt{g}_{\mathrm{in}})$ and $G' = (\Xi, \Sigma, R', \mathtt{g}_{\mathrm{in}})$ where R' is obtained from R by replacing the rule $\mathtt{g}(x_1, \ldots, x_k) \to K$ by all rules $\mathtt{g}(x_1, \ldots, x_k) \to K[v \leftarrow K']$ such that $\mathtt{h}(x_1, \ldots, x_m) \to K'$ is in R; here v is a node of K such that $lab_K(v) = \mathtt{h}^{(m)} \in \Xi$.

We will prove that $\sigma_{\mathcal{N}} = \sigma_{\mathcal{N}'}$, where $\sigma_{\mathcal{N}}$ is the least fixed point of $\varphi_{\mathcal{N}}$ and similarly for $\mathcal{N}'$ (see Definition 5). For $m \in \mathbb{N}$, we will denote $\varphi^m_{\mathcal{N}}(0)$ by $\sigma_{\mathcal{N},m}$, and similarly for $\mathcal{N}'$. Thus, by Proposition 1, $\sigma_{\mathcal{N}}$ is the least upper bound of all $\sigma_{\mathcal{N},m}$, $m \in \mathbb{N}$. Note that by Definition 5, $\sigma_{\mathcal{N},m+1}(\mathtt{k}) = \sigma_{\mathcal{N},m}(\mathrm{rhs}_G(\mathtt{k}))$ for every $m \in \mathbb{N}$ and $\mathtt{k} \in \Xi$; and, of course, $\sigma_{\mathcal{N},0}(\mathtt{k}) = \emptyset$.

We first show that $\sigma_{\mathcal{N}',m} \leq \sigma_{\mathcal{N}}$ for all $m \in \mathbb{N}$, which implies that $\sigma_{\mathcal{N}'} \leq \sigma_{\mathcal{N}}$. It suffices to prove that $\sigma_{\mathcal{N}',m}(\mathtt{k}) \subseteq \sigma_{\mathcal{N}}(\mathtt{k})$ for all $\mathtt{k} \in \Xi$ and $m \in \mathbb{N}$. The proof is by

induction on m. It is trivial for $m = 0$. For the induction step, consider $\sigma_{\mathcal{N}',m+1}(\mathtt{k})$. Since this equals $\sigma_{\mathcal{N}',m}(\mathrm{rhs}_{G'}(\mathtt{k}))$, it remains to prove that $\sigma_{\mathcal{N}',m}(J) \subseteq \sigma_{\mathcal{N}}(\mathtt{k})$ for every $J \in \mathrm{rhs}_{G'}(\mathtt{k})$. We consider two cases.

Case 1: $J \in \mathrm{rhs}_G(\mathtt{k})$. By induction we have $\sigma_{\mathcal{N}',m}(J) \subseteq \sigma_{\mathcal{N}}(J)$; because, in general, if $\tau_1 \leq \tau_2$ then $\tau_1(J) \subseteq \tau_2(J)$. Moreover $\sigma_{\mathcal{N}}(J) \subseteq \sigma_{\mathcal{N}}(\mathtt{k})$, because $\sigma_{\mathcal{N}}$ is a fixed point of $\varphi_{\mathcal{N}}$.

Case 2: $\mathtt{k} = \mathtt{g}$ and $J = K[v \leftarrow K']$. By Lemma 5, $\sigma_{\mathcal{N}',m}(K[v \leftarrow K']) = \sigma_{\mathcal{N}',m}\langle @ := \sigma_{\mathcal{N}',m}(K')\rangle(K[v \leftarrow @])$. By induction, and since $\sigma_{\mathcal{N}}$ is a fixed point of $\varphi_{\mathcal{N}}$, we obtain that $\sigma_{\mathcal{N}',m}(K') \subseteq \sigma_{\mathcal{N}}(\mathtt{h})$. Thus, again by induction,

$$\sigma_{\mathcal{N}',m}(K[v \leftarrow K']) \subseteq \sigma_{\mathcal{N}}\langle @ := \sigma_{\mathcal{N}}(\mathtt{h})\rangle(K[v \leftarrow @]) = \sigma_{\mathcal{N}}(K) \subseteq \sigma_{\mathcal{N}}(\mathtt{g}).$$

In the other direction, we prove by induction on m that $\sigma_{\mathcal{N},m} \leq \sigma_{\mathcal{N}'}$. As above, it suffices to prove in the induction step that $\sigma_{\mathcal{N},m}(J) \subseteq \sigma_{\mathcal{N}'}(\mathtt{k})$ for every $J \in \mathrm{rhs}_G(\mathtt{k})$. As above there are two cases. The first case, where $J \in \mathrm{rhs}_{G'}(\mathtt{k})$, is handled as above. It remains to consider the second case, where $\mathtt{k} = \mathtt{g}$ and $J = K$. If $m = 0$ then $\sigma_{\mathcal{N},m}(K) = \emptyset$ because a non-variable (namely $\mathtt{h}$) occurs in K, and we are ready. Now let $m \geq 1$. Then

$$\begin{array}{rcl}
\sigma_{\mathcal{N},m}(K) & = & \sigma_{\mathcal{N},m}\langle @ := \sigma_{\mathcal{N},m}(\mathtt{h})\rangle(K[v \leftarrow @]) \\
& = & \sigma_{\mathcal{N},m}\langle @ := \sigma_{\mathcal{N},m-1}(\mathrm{rhs}_G(\mathtt{h}))\rangle(K[v \leftarrow @]) \\
& = & \bigcup_{K' \in \mathrm{rhs}_G(\mathtt{h})} \sigma_{\mathcal{N},m}\langle @ := \sigma_{\mathcal{N},m-1}(K')\rangle(K[v \leftarrow @]) \\
& \subseteq & \bigcup_{K' \in \mathrm{rhs}_G(\mathtt{h})} \sigma_{\mathcal{N}'}\langle @ := \sigma_{\mathcal{N}'}(K')\rangle(K[v \leftarrow @]) \\
& = & \bigcup_{K' \in \mathrm{rhs}_G(\mathtt{h})} \sigma_{\mathcal{N}'}(K[v \leftarrow K']) \\
& \subseteq & \sigma_{\mathcal{N}'}(\mathtt{g})
\end{array}$$

where the last three steps are by induction, by Lemma 5, and by the fact that $\sigma_{\mathcal{N}'}$ is a fixed point of $\varphi_{\mathcal{N}'}$, respectively. □

We can now prove that tree delegation networks are as powerful as jungle delegation networks.

Theorem 5. *For every jungle delegation network $\mathcal{N}$ there is a tree delegation network $\mathcal{N}'$ over the same algebra such that $L(\mathcal{N}') = L(\mathcal{N})$.*

Proof. Let $\mathcal{N} = (G, \mathbb{A}, \sigma)$. Then we define $\mathcal{N}' = (H, \mathbb{A}, \sigma)$ where H is the CFTG constructed in the proof of Theorem 2. Since the proof of $L_{\mathrm{J}}(H) = L_{\mathrm{J}}(G)$ was entirely based on Theorem 1, the proof of $L(\mathcal{N}') = L(\mathcal{N})$ is exactly the same, now based on Theorem 4. □

Note that the construction of $\mathcal{N}'$ in the above proof does not depend on the given algebra. Thus, Theorem 5 is a program-schematic result.

Finally, we prove a Mezei-Wright-like result for jungle delegation networks $\mathcal{N} = (G, \mathbb{A}, \sigma)$: the language defined by $\mathcal{N}$ is equal to the semantics of the jungle language generated by G.

Theorem 6. *For every jungle delegation network* $\mathcal{N} = (G, \mathbb{A}, \sigma)$,

$$L(\mathcal{N}) = \sigma(L_{\mathrm{J}}(G)).$$

If, moreover, $(\mathbb{A}, \sigma)$ *is deterministic, then* $L(\mathcal{N}) = \sigma(L_{\mathrm{T}}(G))$.

Proof. Let $\mathcal{N}' = (H, \mathbb{A}, \sigma)$ be the tree delegation network constructed in the proofs of Theorems 2 and 5. Then $L(\mathcal{N}) = L(\mathcal{N}')$ by Theorem 5, and $L_{\mathrm{J}}(G) = L_{\mathrm{J}}(H)$ by Theorem 2. It is proved in [5, Theorem 5.6] for tree delegation networks that $L(\mathcal{N}') = \sigma(L_{\mathrm{J}}(H))$.

In [5, Theorem 6.6] it is proved that if $(\mathbb{A}, \sigma)$ is deterministic, then $L(\mathcal{N}') = \sigma(L_{\mathrm{IO}}(H))$. In the proof of Theorem 3 we already saw that $L_{\mathrm{IO}}(H) = L_{\mathrm{T}}(G)$. □

Example 6. Let G_1 and G_2 be the grammars of Examples 1 and 2. We have seen in Example 2 that $L_{\mathrm{J}}(G_1) = L_{\mathrm{J}}(G_2)$. Hence, by Theorem 6, $L(\mathcal{N}_1) = L(\mathcal{N}_2)$ for all delegation networks $\mathcal{N}_1 = (G_1, \mathbb{A}, \sigma)$ and $\mathcal{N}_2 = (G_2, \mathbb{A}, \sigma)$ over the same algebra $(\mathbb{A}, \sigma)$. In other words, G_1 and G_2 are equivalent program schemes. In particular, by Example 5, we have that $L(\mathcal{N}_1) = \sigma(K'') = \{\diamondsuit\diamondsuit\diamondsuit, \diamondsuit\heartsuit\heartsuit\}$ for the string algebra $(\mathbb{A}, \sigma)$ defined in Example 4.

Note that $L(\mathcal{N}_1)$ is not equal to $\sigma(L_{\mathrm{T}}(G_1))$. In fact, $L_{\mathrm{T}}(G_1) = \{\mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a}))\}$ (see the footnote in the proof of Theorem 3), and $\sigma(\mathtt{f}(\mathtt{a}, \mathtt{f}(\mathtt{a}, \mathtt{a})))$ is the set of all strings of length 3 in $\mathbb{A} = \{\diamondsuit, \heartsuit\}^*$. As discussed in Example 4, this illustrates two effects: in the IO-generated tree the second and third `a` are not shared, so that the second and third symbol of the resulting strings may differ. Morover, the node with label `d` above the root of K'' (which is garbage) is not present, thus allowing the first symbol of the string to differ from $\diamondsuit$. In fact, the node with label `c` is garbage as well and thus not present in the IO-generated tree. If we redefine $\sigma(\mathtt{c}) = \emptyset$, see Example 4, then $L(\mathcal{N}_1) = L(\mathcal{N}_2) = \emptyset$ whereas $\sigma(L_{\mathrm{T}}(G_1))$ is the same as above. □

References

[1] Arbib, M. A. and Give'on, Y. Algebra automata I: Parallel programming as a prolegomena to the categorical approach. *Information and Control*, 12:331–345, 1968.

[2] Ariola, Z. M. and Klop, J. W. Equational term graph rewriting. *Fundamenta Informaticae*, 26:207–240, 1996.

[3] Courcelle, B. An axiomatic definition of context–free rewriting and its application to NLC graph grammars. *Theoretical Computer Science*, 55:141–181, 1987.

[4] Drewes, F. From tree-based generation to delegation networks. In Bozapalidis, S. and Rahonis, G., editors, *Proc. 2nd International Conference on Algebraic Informatics (CAI'07)*, volume 4728 of Lecture Notes in Computer Science, pages 48–72. Springer, 2007.

[5] Drewes, F. and Engelfriet, J. The generative power of delegation networks. Report 12.15, Umeå University, 2012. To appear in *Information and Computation*. Available online doi:10.1016/j.ic.2015.04.005

[6] Drewes, F., Habel, A. and Kreowski, H.-J. Hyperedge replacement graph grammars. In Rozenberg, G., editor, *Handbook of Graph Grammars and Computing by Graph Transformation*. Vol. 1: *Foundations*, chapter 2, pages 95–162. World Scientific, Singapore, 1997.

[7] Duske, J., Parchmann, R., Sedello, M., and Specht, J. IO-macro languages and attributed translations. *Information and Control*, 35:87–105, 1977.

[8] Engelfriet, J. and Filè, G. The formal power of one-visit attribute grammars. *Acta Informatica*, 16:275–302, 1981.

[9] Engelfriet, J. and Heyker, L. Context–free hypergraph grammars have the same term-generating power as attribute grammars. *Acta Informatica*, 29:161–210, 1992.

[10] Engelfriet, J. and Maneth, S. Tree languages generated by context-free graph grammars. In Ehrig, H., Engels, G., and Kreowski, H.-J., editors, *Proc. Theory and Application of Graph Transformations (TAGT'98)*, volume 1764 of *Lecture Notes in Computer Science*, pages 15–29, 2000.

[11] Engelfriet, J. and Schmidt, E. M. IO and OI. *Journal of Computer and System Sciences*, 15:328–353, 1977, and 16:67–99, 1978.

[12] Engelfriet, J. and Vogler, H. The translation power of top-down tree-to-graph transducers. *Journal of Computer and System Sciences*, 49:258–305, 1994.

[13] Gécseg, F. and Steinby, M. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984.

[14] Gécseg, F. and Steinby, M. Tree languages. In Rozenberg, G. and Salomaa, A., editors, *Handbook of Formal Languages.* Vol. 3: *Beyond Words*, chapter 1, pages 1–68. Springer, 1997.

[15] Habel, A., Kreowski, H.-J., and Plump, D. Jungle evaluation. *Fundamenta Informaticae*, 15:37–60, 1991.

[16] Hoffmann, B. and Plump, D. Implementing term rewriting by jungle evaluation. *RAIRO Theoretical Informatics and Applications*, 1991.

[17] Mezei, J. and Wright, J. B. Algebraic automata and context-free sets. *Information and Control*, 11:3–29, 1967.

[18] Plump, D. Term graph rewriting. In Ehrig, H., Engels, G., Kreowski, H.-J., and Rozenberg, G., editors, *Handbook of Graph Grammars and Computing by Graph Transformation.* Vol. 2: *Applications, Languages, and Tools*, chapter 1, pages 3–61. World Scientific, Singapore, 1999.

Received 29th April 2015

Acta Cybernetica **22** (2015) 393–402.

Local Weighted Tree Languages*

Zoltán Fülöp†

To the memory of my teacher and colleague Ferenc Gécseg

Abstract

Local weighted tree languages over semirings are introduced. For an arbitrary semiring, a weighted tree language is shown to be recognizable iff it appears as the image of a local weighted tree language under a deterministic relabeling.

1 Introduction

Trees or terms are fundamental concepts among others in computer science. In this paper we consider trees over ranked alphabets. Tree automata were introduced in the 60s of the last century in [6, 18, 20] and since then the theory of tree automata and tree languages has developed rapidly, see [12, 13] and [5] for surveys. Not much later, already in the 80s, quantitative aspects gained attention and weighted tree automata were introduced in [2, 1]. Within the last decades several authors have dealt with different weighted tree automaton models and their behaviour. Among others, for weighted tree automata over semirings, a Kleene-type characterization was obtained in [7], fixed point characterizations in [16, 4], and a characterization by weighted monadic second-order logic in [8, 9]. A summary of these and several other results on weighted tree automata and weighted tree languages can be found in [10] and [11].

Local tree languages were considered first time in [6, 17, 18, 19]. They are defined in the way that the membership of a tree to a local tree language can be decided by checking local properties of that tree. More exactly, for a ranked alphabet Σ, a Σ-fork (shortly: fork) is a tuple $(\sigma_1 \dots \sigma_k, \sigma)$, where $\sigma \in \Sigma$ is a symbol of arity k and $\sigma_1, \dots, \sigma_k$ are further symbols in Σ. The fork $(\sigma_1 \dots \sigma_k, \sigma)$ occurs in a tree if the tree has a σ-node of which the k sons are labeled by $\sigma_1, \dots, \sigma_k$ from left to right. Let $\mathrm{Fork}(\Sigma)$ be the set of all Σ-forks. Moreover, let us fix a subset $F \subseteq \mathrm{Fork}(\Sigma)$ of admissible forks and a subset $R \subseteq \Sigma$ of admissible roots. Then, a tree $\xi \in T_\Sigma$ belongs to the local tree language determined by the couple (F, R) if and only if all forks in ξ belong to F and the root of ξ belongs to R. A summary

*This work was supported by the NKFI grant K 108 448.

†Department of Foundations of Computer Science, University of Szeged, Árpád tér 2, 6720 Szeged, Hungary. E-mail: fulop@inf.u-szeged.hu

and the main result of these investigations is presented in [12, Sect. II.9] and [13, Sect. 9]. The main result is a characterization of recognizable tree languages by images of local tree languages under deterministic relabelings, cf. [12, Thm. II.9.5] and [13, Prop. 8.1].

To the best of the author's knowledge, the quantitative aspects of local tree languages has not been investigated yet. In this paper we fill this gap in the theory of weighted tree languages. We introduce the concept of a local weighted tree language over a semiring S in a natural way. Namely, we associate a weight to each fork by a mapping $\varphi : \mathrm{Fork}(\Sigma) \to S$ and to each root by another mapping $\rho : \Sigma \to S$. We note that in both cases the weight can be 0. Then the weight of a tree $\xi \in T_\Sigma$ will be the (semiring) product of the weights associated to the forks in ξ and the weight associated to the root of ξ. The order of the factors is the postorder of the nodes of ξ. Finally, we show that that the mentioned characterization result in the classical (unweighted) case can be generalized to the weighted one. In fact, we prove (cf. Theorem 1) that a weighted tree language over an arbitrary semiring is recognizable if and only if it can be obtained as the image of a local weighted tree language under a deterministic relabeling.

2 Preliminaries

We denote by $\mathbb{N}$ the set of nonnegative integers. Let Q and S be sets, and let $k \in \mathbb{N}$. We will write just $q_1 \dots q_k$ for an element $(q_1, \dots, q_k)$ of Q^k. Hence $Q^0 = \{\varepsilon\}$. We denote the set of all mappings $v : Q \to S$ by S^Q. For each $v \in S^Q$ and $q \in Q$, we abbreviate $v(q)$ by v_q.

A *ranked alphabet* is a tuple (Σ, rk) where Σ is a finite set and $rk : \Sigma \to \mathbb{N}$ is a mapping called rank mapping. For every $k \geq 0$, we define $\Sigma_k = \{\sigma \in \Sigma \mid rk(\sigma) = k\}$. Sometimes we write $\sigma^{(k)}$ to mean that $\sigma \in \Sigma_k$. Moreover, let H be a set disjoint with Σ. The set of Σ-*terms over* H, denoted by $T_\Sigma(H)$, is the smallest set T such that (i) $\Sigma_0 \cup H \subseteq T$ and (ii) if $k \geq 1$, $\sigma \in \Sigma_k$, and $\xi_1, \dots, \xi_k \in T$, then $\sigma(\xi_1, \dots, \xi_k) \in T$. We denote $T_\Sigma(\emptyset)$ by T_Σ.

We define the *height* and the *root* of trees as the functions $\mathrm{height} : T_\Sigma \to \mathbb{N}$ and $\mathrm{rt} : T_\Sigma \to \Sigma$, respectively, as follows: (i) for every $\alpha \in \Sigma_0$, we define $\mathrm{height}(\alpha) = 0$, $\mathrm{rt}(\alpha) = \alpha$ and (ii) for every $\xi = \sigma(\xi_1, \dots, \xi_k)$, where $k \geq 1$, we define $\mathrm{height}(\xi) = 1 + \max\{\mathrm{height}(\xi_i) \mid 1 \leq i \leq k\}$ and $\mathrm{rt}(\xi) = \sigma$.

A *semiring* $(S, +, \cdot, 0, 1)$ is an algebra which consists of a commutative monoid $(S, +, 0)$, called the additive monoid of S, and a monoid $(S, \cdot, 1)$, called the multiplicative monoid of S, such that multiplication distributes (from both left and right) over addition, and moreover, $0 \neq 1$ and 0 is absorbing with respect to $\cdot$ (also both from left and right). An introduction to and several details about semirings can be found e.g. in the books [14] and [15].

> *In the rest of this paper Σ and Δ will denote arbitrary ranked alphabets unless specified otherwise, and S will denote an arbitrary semiring.*

A *deterministic relabeling* (for short: drel) is a mapping $\tau : \Sigma \to \Delta$ satisfying

$\tau(\Sigma_k) \subseteq \Delta_k$ for every $k \geq 0$. The mapping τ extends to the tree transformation $\tau' : T_\Sigma \to T_\Delta$ defined by $\tau'(\sigma(\xi_1, \ldots, \xi_k)) = \tau(\sigma)(\tau'(\xi_1), \ldots, \tau'(\xi_k))$ for every $k \geq 0$, $\sigma \in \Sigma_k$, and $\xi_1, \ldots, \xi_k \in T_\Sigma$. In what follows we will call τ' also a drel and write τ for τ'.

A *weighted tree language over* Σ *and* S (for short: weighted tree language) is a mapping $\Phi : T_\Sigma \to S$, and a *weighted tree language over* S is a weighted tree language over Σ and S for some ranked alphabet Σ. For every $\xi \in T_\Sigma$, the element $\Phi(\xi)$ of S is called the *weight* of ξ. Now let $\tau : T_\Sigma \to T_\Delta$ be a drel. We extend τ to weighted tree languages as follows: for every $\Phi : T_\Sigma \to S$, we define $\tau(\Phi) : T_\Delta \to S$ by

$$\tau(\Phi)(\zeta) = \sum_{\xi \in T_\Sigma, \tau(\xi) = \zeta} \Phi(\xi)$$

for every $\zeta \in T_\Delta$. Let $\mathrm{C}(S)$ be a class of weighted tree languages over S. We denote by d-REL($\mathrm{C}(S)$) the class of all weighted tree languages $\tau(\Phi)$, where τ is a drel and $\Phi \in \mathrm{C}(S)$.

A Σ-*algebra* (V, θ) consists of a nonempty set V (*carrier set*) and an arity preserving mapping θ, called the *interpretation*, from Σ to the set operations over V, i.e., $\theta(\sigma) : V^k \to V$ for every $k \geq 0$ and $\sigma \in \Sigma_k$. The Σ-*term algebra* (T_Σ, top), defined by $\mathrm{top}(\sigma)(\xi_1, \ldots, \xi_k) = \sigma(\xi_1, \ldots, \xi_k)$ for $\sigma \in \Sigma_k$ and $\xi_1, \ldots, \xi_k \in T_\Sigma$, is *initial* in the class of all Σ-algebras, i.e., for every Σ-algebra (V, θ), there is a unique Σ-algebra homomorphism from T_Σ to V.

A *weighted tree automaton (over* Σ *and* S*)* (for short: wta) is a tuple $\mathcal{A} = (Q, \Sigma, S, \delta, \kappa)$ where

- Q is a finite nonempty set, the *set of states*,
- Σ is the *ranked input alphabet*,
- $\delta = (\delta_k \mid k \in \mathbb{N})$ is a *family of transition mappings*[1] $\delta_k : Q^k \times \Sigma_k \times Q \to S$,
- $\kappa : Q \to S$ is the *root weight mapping.*

For every $k \in \mathbb{N}$ and transition $w = (q_1 \ldots q_k, \sigma, q) \in Q^k \times \Sigma_k \times Q$, and $1 \leq i \leq k$, we call q_i the ith input state of w and denote it by $\mathrm{in}_i(w)$. Similarly, we call q the output state of w and denote it by $\mathrm{out}(w)$. Moreover, we call the element $\delta_k(w)$ of S the weight of the transition w.

For $\mathcal{A}$ we consider the Σ-algebra $(S^Q, \delta_\mathcal{A})$ where, for every $k \geq 0$ and $\sigma \in \Sigma_k$, the k-ary operation $\delta_\mathcal{A}(\sigma) : S^Q \times \ldots \times S^Q \to S^Q$ is defined by

$$\delta_\mathcal{A}(\sigma)(v_1, \ldots, v_k)_q = \sum_{q_1, \ldots, q_k \in Q} (v_1)_{q_1} \cdot \ldots \cdot (v_k)_{q_k} \cdot \delta_k(q_1 \ldots q_k, \sigma, q)$$

for every $q \in Q$ and $v_1, \ldots, v_k \in S^Q$. (Here $\sum$ and $\cdot$ denote a finite sum and the multiplication in the semiring S, respectively.) Let us denote the unique Σ-algebra

[1] In the literature δ is also called a *tree representation* and δ_k is given as a mapping of type $\Sigma_k \to S^{Q^k \times Q}$.

homomorphism from T_Σ to S^Q by h_δ. The weighted tree language $\|\mathcal{A}\| : T_\Sigma \to S$ *recognized* by $\mathcal{A}$ is defined by

$$\|\mathcal{A}\|(\xi) = \sum_{q \in Q} h_\delta(\xi)_q \cdot \kappa(q)$$

for every $\xi \in T_\Sigma$. Due to the definitions of $\delta_\mathcal{A}$ and h_δ, we obtain that

$$h_\delta(\sigma(\xi_1, \ldots, \xi_k))_q = \sum_{q_1, \ldots, q_k \in Q} h_\delta(\xi_1)_{q_1} \cdot \ldots \cdot h_\delta(\xi_k)_{q_k} \cdot \delta_k(q_1 \ldots q_k, \sigma, q) \qquad (1)$$

for every $\sigma(\xi_1, \ldots, \xi_k) \in T_\Sigma$ and $q \in Q$. An introduction to the theory of wta over semirings and several results can be found in [10] and [11].

Example 1. (Cf. [3, Example 3.3]) We consider the arctic semiring $\mathsf{Arct} = (\mathbb{N} \cup \{-\infty\}, \max, +, -\infty, 0)$ and construct the wta $\mathcal{A} = (Q, \Sigma, \mathsf{Arct}, \delta, \kappa)$ which recognizes the weighted tree language height. Let $Q = \{p_1, p_2\}$, $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}\}$, and $\kappa(p_1) = 0$ and $\kappa(p_2) = -\infty$. Moreover, let

$$\begin{array}{lcl} \delta_0(\varepsilon, \alpha, p_1) = \delta_0(\varepsilon, \alpha, p_2) = 0, \\ \delta_2(p_1p_2, \sigma, p_1) = \delta_2(p_2p_1, \sigma, p_1) = 1, \\ \delta_2(p_2p_2, \sigma, p_2) = 0, \end{array}$$

and for every other transition (q_1q_2, σ, q) we have $\delta_2(q_1q_2, \sigma, q) = -\infty$. We consider the tree $\xi = \sigma(\alpha, \alpha)$ and compute $h_\delta(\xi)_{p_1}$ and $h_\delta(\xi)_{p_2}$. Clearly, $h_\delta(\alpha)_{p_1} = \delta_0(\alpha)_{\varepsilon, p_1} = 0$ and $h_\delta(\alpha)_{p_2} = 0$. Then

$$h_\delta(\sigma(\alpha, \alpha))_{p_1} = \max_{q_1, q_2 \in Q}\{h_\delta(\alpha)_{q_1} + h_\delta(\alpha)_{q_2} + \delta_2(q_1q_2, \sigma, p_1)\} = 1$$

(note that $\delta_2(p_1p_1, \sigma, p_1) = \delta_2(p_2p_2, \sigma, p_1) = -\infty$ and $-\infty$ is neutral for max) and, similarly, $h_\delta(\sigma(\alpha, \alpha))_{p_2} = 0$. In general, we can prove by structural induction on ξ that $h_\delta(\xi)_{p_1} = \mathrm{height}(\xi)$ and $h_\delta(\xi)_{p_2} = 0$ for every $\xi \in T_\Sigma$. Thus $\|\mathcal{A}\| = \mathrm{height}$ and hence $\mathrm{height} \in \mathrm{Rec}(\Sigma, \mathsf{Arct})$.

A wta $\mathcal{A} = (Q, \Sigma, S, \delta, \kappa)$ is *bottom-up deterministic* (for short: bu-deterministic) if for every $k \geq 0$, $\sigma \in \Sigma_k$, and $w \in Q^k$ there is at most one $q \in Q$ such that $\delta_k(w, \sigma, q) \neq 0$. If $\mathcal{A}$ is bu-deterministic, then for every input tree $\xi \in T_\Sigma$, there is at most one $q \in Q$ such that $h_\delta(\xi)_q \neq 0$. In this case the operation $+$ of S is not used for the computation of $\|\mathcal{A}\|$.

A weighted tree language $\Phi : T_\Sigma \to S$ is *recognizable* (resp. bu-*deterministically recognizable*) if there is a wta (resp. bu-deterministic wta) $\mathcal{A}$ such that $\Phi = \|\mathcal{A}\|$. The class of all recognizable weighted tree languages over Σ and S (resp. over S) is denoted by $\mathrm{Rec}(\Sigma, S)$ (resp. $\mathrm{Rec}(S)$). The notation bud-$\mathrm{Rec}(\Sigma, S)$ is introduced analogously.

Finally, we recall that recognizable weighted tree languages are closed under (deterministic) relabelings. A proof can be found, e.g., in [8, Lm. 3.4]. However, in [8] a wta is defined over a commutative semiring and the semantics of a wta is defined in terms of runs. Therefore we give a short proof for our case.

Proposition 1. d-REL$(\mathrm{Rec}(S)) \subseteq \mathrm{Rec}(S)$.

Proof. Let $\mathcal{A} = (Q, \Sigma, S, \delta, \kappa)$ be a wta and $\tau : T_\Sigma \to T_\Delta$ be a drel. We define the wta $\mathcal{A}' = (Q, \Delta, S, \delta', \kappa)$ by

$$\delta'(q_1 \dots q_k, \omega, q) = \sum_{\sigma \in \Sigma_k, \tau(\sigma) = \omega} \delta(q_1 \dots q_k, \sigma, q)$$

for every $k \geq 0$, $\omega \in \Delta_k$, and $q, q_1, \dots, q_k \in Q$, and show that $\mathcal{A}'$ computes $\tau\big(\|\mathcal{A}\|\big)$. We can show by induction on the height of ζ and using equality (1) that

$$h_{\delta'}(\zeta)_q = \sum_{\xi \in T_\Sigma, \tau(\xi) = \zeta} h_\delta(\xi)_q$$

for every $\zeta \in T_\Delta$ and $q \in Q$. Then we get

$$\|\mathcal{A}\|'(\zeta) = \sum_{q \in Q} h_{\delta'}(\zeta)_q \cdot \kappa(q) = \sum_{q \in Q} \Big(\sum_{\xi \in T_\Sigma, \tau(\xi) = \zeta} h_\delta(\xi)_q \Big) \cdot \kappa(q) =$$
$$\sum_{\xi \in T_\Sigma, \tau(\xi) = \zeta} \Big(\sum_{q \in Q} h_\delta(\xi)_q \cdot \kappa(q) \Big) = \sum_{\xi \in T_\Sigma, \tau(\xi) = \zeta} \|\mathcal{A}\|(\xi) = \tau\big(\|\mathcal{A}\|\big)(\zeta)$$

for each $\zeta \in T_\Delta$, which proves $\|\mathcal{A}\|' = \tau\big(\|\mathcal{A}\|\big)$. □

3 The result

A Σ-*fork* (shortly: fork) is a tuple $(\sigma_1 \dots \sigma_k, \sigma)$ for some $k \geq 0$, where $\sigma \in \Sigma_k$ and $\sigma_1, \dots, \sigma_k$ are further symbols in Σ. The fork $(\sigma_1 \dots \sigma_k, \sigma)$ occurs in a tree if the tree has a σ-node of which the k sons are labeled by $\sigma_1, \dots, \sigma_k$ from left to right. We consider the family $\mathrm{Fork}(\Sigma) = (\mathrm{Fork}_k(\Sigma) \mid k \geq 0)$, where

$$\mathrm{Fork}_k(\Sigma) = \{(\sigma_1 \dots \sigma_k, \sigma) \mid \sigma_1, \dots, \sigma_k \in \Sigma, \sigma \in \Sigma_k\}.$$

Note that $\mathrm{Fork}_k(\Sigma) = \Sigma^k \times \Sigma_k$, hence $\mathrm{Fork}_0(\Sigma) = \Sigma_0$.

A *weighted local system (over Σ and S)* (for short: wls) is a pair $\mathcal{L} = (\Sigma, S, \varphi, \rho)$, where φ is a family of mappings $(\varphi_k \mid k \geq 0)$ with $\varphi_k : \mathrm{Fork}_k(\Sigma) \to S$ and $\rho : \Sigma \to S$ is a further mapping. Intuitively, we associate a weight, i.e., an element of S to each fork and also to each symbol in Σ. Note that this weight can be 0.

Next we define the weighted tree language determined by $\mathcal{L}$. For this, we extend φ to the mapping $\varphi' : T_\Sigma \to S$ defined by induction as follows:

(i) $\varphi'(\sigma) = \varphi_0(\sigma)$ for every $\sigma \in \Sigma_0$,

(ii) $\varphi'(\sigma(\xi_1, \dots, \xi_k)) = \varphi'(\xi_1) \cdot \ldots \cdot \varphi'(\xi_k) \cdot \varphi_k(\mathrm{rt}(\xi_1) \dots \mathrm{rt}(\xi_k), \sigma)$ for every $k \geq 1$, $\sigma \in \Sigma_k$, and $\xi_1, \dots, \xi_k \in T_\Sigma$.

In the following we write φ for φ'. The *weighted tree language* $\|\mathcal{L}\| : T_\Sigma \to S$ *determined by* $\mathcal{L}$ is defined by $\|\mathcal{L}\|(\xi) = \varphi(\xi) \cdot \rho(\mathrm{rt}(\xi))$ for every $\xi \in T_\Sigma$. Note that, like for deterministic wta, the operation $+$ of S is not used for the definition of $\|\mathcal{L}\|$.

Thus, $\varphi(\xi)$ is the (semiring) product of the weights associated to the forks in ξ. The order of the factors is the postorder of the nodes of ξ. Moreover, the weight $\|\mathcal{L}\|(\xi)$ of ξ is the product of $\varphi(\xi)$ and the weight associated to the root of ξ.

A weighted tree language $\Phi : T_\Sigma \to S$ is called *local* if there is a wls $\mathcal{L}$ such that $\Phi = \|\mathcal{L}\|$. The class of all local weighted tree languages over Σ and S (resp. over S) is denoted by $\mathrm{Loc}(\Sigma, S)$ (resp. $\mathrm{Loc}(S)$).

Example 2. We consider again the ranked alphabet $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}\}$. We define the wls $\mathcal{L} = (\Sigma, \mathsf{Arct}, \varphi, \rho)$ by

- $\varphi_2(\sigma\alpha, \sigma) = \varphi_2(\alpha\alpha, \sigma) = 1$ and in every other case $\varphi_2(\_\_, \sigma) = 0$,
- $\varphi_0(\varepsilon, \alpha) = 0$, and by $\rho(\sigma) = \rho(\alpha) = 0$.

It should be clear that $\|\mathcal{L}\|(\xi)$ is the number of the occurrences of the pattern $\sigma(\_, \alpha)$ in ξ, where '_' is a placeholder which may be filled by either σ or α. We note that in [11, Example 3.4] a wta is given over the semiring of natural numbers which recognizes $\|\mathcal{L}\|$.

Next we show that local weighted tree languages are bu-deterministically recognizable.

Lemma 1. $\mathrm{Loc}(\Sigma, S) \subseteq \text{bud-Rec}(\Sigma, S)$.

Proof. Let $\mathcal{L} = (\Sigma, S, \varphi, \rho)$ by a wls over Σ and S. We construct a wta $\mathcal{A} = (Q, \Sigma, S, \delta, \kappa)$ such that $\|\mathcal{A}\| = \|\mathcal{L}\|$. For this, we define

- $Q = \{\overline{\sigma} \mid \sigma \in \Sigma\}$,
- for every $k \geq 0$, $\sigma_1 \dots \sigma_k \in \Sigma^k$, $\sigma \in \Sigma_k$, and $\omega \in \Sigma$,
$$\delta_k(\overline{\sigma_1} \dots \overline{\sigma_k}, \sigma, \overline{\omega}) = \begin{cases} \varphi_k(\sigma_1 \dots \sigma_k, \sigma) & \text{if } \omega = \sigma \\ 0 & \text{otherwise,} \end{cases}$$
- $\kappa(\overline{\sigma}) = \rho(\sigma)$ for every $\sigma \in \Sigma$.

It is clear that $\mathcal{A}$ is bu-deterministic. Next we show the following statement by induction on ξ: for every $\xi \in T_\Sigma$ and $\omega \in \Sigma$, we have

$$h_\delta(\xi)_{\overline{\omega}} = \begin{cases} \varphi(\xi) & \text{if } \omega = \mathrm{rt}(\xi) \\ 0 & \text{otherwise.} \end{cases}$$

Let $\xi = \sigma(\xi_1, \dots, \xi_k)$ for some $k \geq 0$, $\sigma \in \Sigma_k$, and $\xi_1, \dots, \xi_k \in T_\Sigma$. We have

$$h_\delta(\sigma(\xi_1, \dots, \xi_k))_{\overline{\omega}} =$$

$$\sum_{\sigma_1, \dots, \sigma_k \in \Sigma} h_\delta(\xi_1)_{\overline{\sigma_1}} \cdot \ldots \cdot h_\delta(\xi_k)_{\overline{\sigma_k}} \cdot \delta_k(\overline{\sigma_1} \dots \overline{\sigma_k}, \sigma, \overline{\omega}) =$$

$$\varphi(\xi_1) \cdot \ldots \cdot \varphi(\xi_k) \cdot \delta_k(\overline{\mathrm{rt}(\xi_1)} \dots \overline{\mathrm{rt}(\xi_k)}, \sigma, \overline{\omega}) =$$

$$\varphi(\xi_1) \cdot \ldots \cdot \varphi(\xi_k) \cdot \varphi_k(\mathrm{rt}(\xi_1) \dots \mathrm{rt}(\xi_k), \sigma) \text{ if } \omega = \sigma \text{ and } 0 \text{ otherwise} =$$

$$\varphi(\sigma(\xi_1, \dots, \xi_k)) \text{ if } \omega = \sigma \text{ and } 0 \text{ otherwise,}$$

where the second equality is justified by the I. H. and the third one by the definition of δ_k. Note that the case $k = 0$ covers also the base of the induction. Finally, let $\xi \in T_\Sigma$. Then we get

$$\|\mathcal{A}\|(\xi) = \sum_{\omega \in \Sigma} h_\delta(\xi)_{\overline{\omega}} \cdot \kappa(\overline{\omega}) = \varphi(\xi) \cdot \kappa(\overline{\mathrm{rt}(\xi)}) = \varphi(\xi) \cdot \rho(\mathrm{rt}(\xi)) = \|\mathcal{L}\|(\xi),$$

where the second equality follows from the statement and the other ones from the corresponding definitions. □

One can easily find a semiring S and a ranked alphabet Σ such that the inclusion in Lemma 1 is strict. For instance, consider the Boolean semiring $\mathbb{B} = (\{0,1\}, \vee, \wedge, , 0, 1)$, the ranked alphabet $\Sigma = \{\gamma^{(1)}, \alpha^{(0)}\}$ and the weighted tree language Φ defined by $\Phi\big(\gamma(\gamma(\alpha))\big) = 1$ and $\Phi(\xi) = 0$ for every other $\xi \in T_\Sigma$. It is easy to show that $\Phi \in (\text{bud-Rec}(\Sigma, \mathbb{B}) \setminus \text{Loc}(\Sigma, \mathbb{B}))$. Another example can be found for the Boolean case on [12, p. 107].

Finally, we give a characterization of recognizable weighted tree languages by images of local weighted tree languages under deterministic relabelings.

Theorem 1. $\text{Rec}(S) = \text{d-REL}(\text{Loc}(S))$.

Proof. The inclusion from right to left follows from Proposition 1 and Lemma 1. Therefore, we prove the other inclusion.

For this, let $\mathcal{A} = (Q, \Sigma, S, \delta, \kappa)$ be a wta. We will construct a ranked alphabet Δ, a wls $\mathcal{L} = (\Delta, S, \varphi, \rho)$, and a deterministic relabeling $\tau : T_\Delta \to T_\Sigma$ such that $\|\mathcal{A}\| = \tau(\|\mathcal{L}\|)$.

Let $\Delta_k = Q^k \times \Sigma_k \times Q$ for every $k \geq 0$. Moreover, let us define φ and ρ as follows. For every $k \geq 0$, $\omega_1 \dots \omega_k \in \Delta$ and $(q_1 \dots q_k, \sigma, q) \in \Delta_k$, let

$$\varphi_k\big(\omega_1 \dots \omega_k, (q_1 \dots q_k, \sigma, q)\big) = \begin{cases} \delta_k(q_1 \dots q_k, \sigma, q) & \text{if } \mathrm{out}(\omega_i) = q_i \\ & \text{for all } 1 \leq i \leq k \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\rho((q_1 \dots q_k, \sigma, q)) = \kappa(q).$$

Finally, let $\tau_k : \Delta_k \to \Sigma_k$ be defined by $\tau((q_1 \dots q_k, \sigma, q)) = \sigma$ for every $k \geq 0$ and $(q_1 \dots q_k, \sigma, q) \in \Delta_k$.

First we prove the following statement by induction: for every $\xi \in T_\Sigma$ and $q \in Q$, we have

$$h_\delta(\xi)_q = \sum_{\substack{\zeta \in T_\Delta, \tau(\zeta) = \xi \\ \mathrm{out}(\mathrm{rt}(\zeta)) = q}} \varphi(\zeta).$$

Let $\xi = \sigma(\xi_1, \dots, \xi_k)$ for some $k \geq 0$, $\sigma \in \Sigma_k$, and $\xi_1, \dots, \xi_k \in T_\Sigma$. In the following computation we abbreviate a product of the form $a_1 \cdot \ldots \cdot a_k$ by $\prod_{i=1}^k a_i$, where

$a_1, \ldots, a_k \in S$. Then

$$h_\delta(\sigma(\xi_1, \ldots, \xi_k))_q =$$

$$\sum_{q_1,\ldots,q_k \in Q} \Big(\prod_{i=1}^{k} h_\delta(\xi_i)_{q_i} \Big) \cdot \delta_k(q_1 \ldots q_k, \sigma, q) =$$

$$\sum_{q_1,\ldots,q_k \in Q} \Big(\prod_{i=1}^{k} \Big(\sum_{\substack{\zeta_i \in T_\Delta, \tau(\zeta_i) = \xi_i \\ \mathrm{out}(\mathrm{rt}(\zeta_i)) = q_i}} \varphi(\zeta_i) \Big) \Big) \cdot \delta_k(q_1 \ldots q_k, \sigma, q) =$$

$$\sum_{q_1,\ldots,q_k \in Q} \Big(\sum_{\substack{\forall 1 \le i \le k : \zeta_i \in T_\Delta, \\ \tau(\zeta_i) = \xi_i, \mathrm{out}(\mathrm{rt}(\zeta_i)) = q_i}} \prod_{i=1}^{k} \varphi(\zeta_i) \Big) \cdot \delta_k(q_1 \ldots q_k, \sigma, q) =$$

$$\sum_{q_1,\ldots,q_k \in Q} \Big(\sum_{\substack{\forall 1 \le i \le k : \zeta_i \in T_\Delta, \\ \tau(\zeta_i) = \xi_i, \mathrm{out}(\mathrm{rt}(\zeta_i)) = q_i}} \big(\prod_{i=1}^{k} \varphi(\zeta_i) \big) \cdot \delta_k(q_1 \ldots q_k, \sigma, q) \Big) =$$

$$\sum_{\substack{\forall 1 \le i \le k : \zeta_i \in T_\Delta, \\ \tau(\zeta_i) = \xi_i}} \big(\prod_{i=1}^{k} \varphi(\zeta_i) \big) \cdot \delta_k\big(\mathrm{out}(\mathrm{rt}(\zeta_1)) \ldots \mathrm{out}(\mathrm{rt}(\zeta_k)), \sigma, q \big) =$$

$$\sum_{\substack{\forall 1 \le i \le k : \zeta_i \in T_\Delta, \\ \tau(\zeta_i) = \xi_i}} \big(\prod_{i=1}^{k} \varphi(\zeta_i) \big) \cdot \varphi_k\big(\mathrm{rt}(\zeta_1) \ldots \mathrm{rt}(\zeta_k), \big(\mathrm{out}(\mathrm{rt}(\zeta_1)) \ldots \mathrm{out}(\mathrm{rt}(\zeta_k)), \sigma, q \big) \big) =$$

$$\sum_{\substack{\forall 1 \le i \le k : \zeta_i \in T_\Delta, \\ \tau(\zeta_i) = \xi_i, q_i \in Q}} \big(\prod_{i=1}^{k} \varphi(\zeta_i) \big) \cdot \varphi_k\big(\mathrm{rt}(\zeta_1) \ldots \mathrm{rt}(\zeta_k), (q_1 \ldots q_k, \sigma, q) \big) =$$

$$\sum_{\substack{\forall 1 \le i \le k : \zeta_i \in T_\Delta, \\ \tau(\zeta_i) = \xi_i, q_i \in Q}} \varphi\big((q_1 \ldots q_k, \sigma, q)(\zeta_1, \ldots \zeta_k) \big) =$$

$$\sum_{\substack{\zeta \in T_\Delta, \tau(\zeta) = \sigma(\xi_1,\ldots,\xi_k) \\ \mathrm{out}(\mathrm{rt}(\zeta)) = q}} \varphi(\zeta).$$

The first, second, and the sixth equality follows from (1), the I. H., and the definition of φ, respectively. Finally, the seventh one follows from the fact that if $q_i \neq \mathrm{out}(\mathrm{rt}(\zeta_i))$ for some $1 \le i \le k$, then $\varphi_k\big(\mathrm{rt}(\zeta_1) \ldots \mathrm{rt}(\zeta_k), (q_1 \ldots q_k, \sigma, q)\big) = 0$.

Finally, for every $\xi \in T_\Sigma$, we have

$$\|\mathcal{A}\|(\xi) = \sum_{q\in Q} h_\delta(\xi)_q \cdot \kappa(q) = \sum_{q\in Q} \Bigg(\sum_{\substack{\zeta\in T_\Delta, \tau(\zeta)=\xi \\ \mathrm{out}(\mathrm{rt}(\zeta))=q}} \varphi(\zeta)\cdot\kappa(q) \Bigg) =$$

$$\sum_{\zeta\in T_\Delta,\tau(\zeta)=\xi} \varphi(\zeta)\cdot\kappa(\mathrm{out}(\mathrm{rt}(\zeta))) = \sum_{\zeta\in T_\Delta,\tau(\zeta)=\xi} \varphi(\zeta)\cdot\rho(\mathrm{rt}(\zeta)) =$$

$$\sum_{\zeta\in T_\Delta,\tau(\zeta)=\xi} \|\mathcal{L}\|(\zeta) = \tau(\|\mathcal{L}\|)(\xi),$$

where the second equality is justified by the statement proved above.

□

References

[1] A. Alexandrakis and S. Bozapalidis, Weighted grammars and Kleene's theorem. *Information Processing Letters*, 24(1):1–4, January 1987.

[2] J. Berstel and C. Reutenauer, Recognizable formal power series on trees. *Theoret. Comput. Sci.*, 18(2):115–148, 1982.

[3] B. Borchardt, A Pumping Lemma and Decidability Problems for Recognizable Tree Series, *Acta Cyb.*, 16(4):509–544, 2004.

[4] S. Bozapalidis. Equational elements in additive algebras. *Theory of Computing Systems*, 32(1):1–33, 1999.

[5] H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi, Tree Automata Techniques and Applications. Available on: http://www.grappa.univ-lille3.fr/tata, 1997.

[6] J. Doner, Tree acceptors and some of their applications, *J. Comput. System Sci.*, 4 (1970) 406–451.

[7] M. Droste, Chr. Pech, and H. Vogler. A Kleene theorem for weighted tree automata. *Theory of Computing Systems*, 38:1–38, 2005.

[8] M. Droste and H. Vogler, Weighted tree automata and weighted logics. *Theoret. Comput. Sci.*, 366:228–247, 2006.

[9] M. Droste and H. Vogler, Weighted logics for unranked tree automata. *Theory of Computing Systems*, 48(1):23–47, 2011. published online first, 29. June 2009, doi:10.1007/s00224-009-9224-4.

[10] Z. Ésik and W. Kuich. Formal tree series. *J. of Automata, Languages, and Combinatorics*, 8(2):219–285, 2003.

[11] Z. Fülöp and H. Vogler, Weighted tree automata and tree transducers, *In: Handbook of Weighted Automata (eds. M. Droste, W. Kuich, and H. Vogler)*, chapter 9, pages 313–403, Springer-Verlag, 2009.

[12] F. Gécseg and M. Steinby, *Tree Automata*, Akadémiai Kiadó, 1984.

[13] F. Gécseg and M. Steinby, Tree languages, *in: Handbook of Formal Languages (eds. G. Rozenberg and A. Salomaa)*, Vol. 1, pp.1-68, Springer-Verlag, 1997.

[14] J.S. Golan, *Semirings and their Applications*, Kluwer Academic Publishers, Dordrecht, 1999.

[15] U. Hebisch and H.J. Weinert, *Semirings - Algebraic Theory and Applications in Computer Science*, World Scientific, Singapore, 1998.

[16] W. Kuich. Formal power series over trees. In S. Bozapalidis, editor, *3rd International Conference on Developments in Language Theory, DLT 1997, Thessaloniki, Greece, Proceedings*, pages 61–101. Aristotle University of Thessaloniki, 1998.

[17] M. Takahashi, Generalizations of regular sets and their application to a study of context-free languages. -IC 27 (1975), 1-36.

[18] J.W. Thatcher, Characterizing derivation trees of context-free grammars through a generalization of finite automata theory, *J. Comput. Syst. Sci.*, 1 (1967) 317-322.

[19] J.W. Thatcher, Generalized^2 sequential machine maps, *J. Comput. Syst. Sci.*, 4 (1970) 339-367.

[20] J.W. Thatcher and J.B. Wright, Generalized finite automata theory with an application to a decision problem of second-order logic, *Math. Syst. Theory*, 2 (1968), 57-81.

Received 15th June 2015

Acta Cybernetica **22** (2015) 403–422.

State Complexity of Kleene-Star Operations on Regular Tree Languages*

Yo-Sub Han†, Sang-Ki Ko†, Xiaoxue Piao‡, and Kai Salomaa‡

Dedicated to the memory of Professor Ferenc Gécseg (1939–2014)

Abstract

The concatenation of trees can be defined either as a sequential or a parallel operation, and the corresponding iterated operation gives an extension of Kleene-star to tree languages. Since the sequential tree concatenation is not associative, we get two essentially different iterated sequential concatenation operations that we call the bottom-up star and top-down star operation, respectively. We establish that the worst-case state complexity of bottom-up star is $(n + \frac{3}{2}) \cdot 2^{n-1}$. The bound differs by an order of magnitude from the corresponding result for string languages. The state complexity of top-down star is similar as in the string case. We consider also the state complexity of the star of the concatenation of a regular tree language with the set of all trees.

Keywords: tree automata, state complexity, iterated concatenation

1 Introduction

The descriptional complexity of finite automata has been studied for over half a century [13, 15, 16], and there has been particularly much work done over the last two decades. The reader may find more information in the surveys [4, 8, 12]. Also the state complexity of various extensions of finite automata, such as tree automata [14, 19] and input-driven pushdown automata (a.k.a. nested word automata) [7, 17] has been considered. These models retain the feature of finite

*A preliminary version of parts of this paper appeared in the proceedings of *Computation, Physics and Beyond, International Workshop on Theoretical Computer Science, WTSC2012.* Han and Ko were supported by the Basic Science Research Program through NRF funded by MEST (2012R1A1A2044562) and the International Cooperation Program managed by NRF of Korea (2014K2A1A2048512). Piao and Salomaa were supported by the Natural Sciences and Engineering Research Council of Canada Grant OGP0147224.

†Department of Computer Science, Yonsei University, Seoul 120-749, Republic of Korea, E-mail: {emmous, narame7}@cs.yonsei.ac.kr

‡School of Computing, Queen's University, Kingston, Ontario K7L 2N8, Canada, E-mail: {piao,ksalomaa}@cs.queensu.ca

automata that a nondeterministic automaton can be converted to an equivalent deterministic automaton.

Concatenation of tree languages can be defined either as a sequential or a parallel operation. Tight state complexity bounds for the concatenation of regular (respectively, subtree-free) tree languages were given in [18] (respectively, [3]) and the state complexity of concatenation operations with the set of all trees was considered in [11].

Here we consider the iterated concatenation of trees, that is, an extension of the Kleene-star operation for tree languages. If defined in the usual way, the iterated parallel concatenation is not a regularity preserving operation and Gécseg and Steinby [6] define the Kleene-star of tree languages slightly differently. Since sequential concatenation of tree languages is non-associative, there are two essentially different ways to define the corresponding iterated operation. We name these variants the *bottom-up star* and the *top-down star* operations. It is easy to see that the top-down (sequential) star operation coincides with the iterated product (Kleene-star) based on parallel concatenation considered in [6].

We give tight state complexity bounds for both the bottom-up and the top-down Kleene-star operations. We show that the bottom-up star of a tree language recognized by a deterministic bottom-up automaton with n states can be recognized by an automaton with $(n + \frac{3}{2}) \cdot 2^{n-1}$ states and, furthermore, there exist worst-case examples where this number of states is needed. This bound is, roughly, n times the corresponding bound for regular string languages. On the other hand, the state complexity of the top-down star operation is shown to coincide with the state complexity of Kleene-star on string languages.

The state complexity of combined operations on regular languages was first considered by A. Salomaa et al. [21], and later there has been much interest in this topic [2, 10]. In the last section we consider the state complexity of tree concatenation combined with star in the special case where one of the argument languages consists of the set of all trees. For some of the combined operations we get tight bounds that are significantly lower than the function composition of the state complexity of concatenation with F_Σ and the state complexity of the corresponding star operation.

To conclude the introduction we comment on the difference between classical ranked tree automata [5] and unranked tree automata. Much of the recent work on tree automata uses automata operating on unranked trees that are used in modern applications such as XML document processing [1, 18, 19, 22]. The transitions of an unranked tree automaton A are defined in terms of regular languages, called *horizontal languages.* Each horizontal language is specified by a deterministic finite automaton (DFA) that processes strings of states of the bottom-up computation, or *vertical states.* The size of A is defined to be the sum of the number of vertical states and the numbers of states of the DFAs used to define the horizontal languages.

In the case of the Kleene-star operations, the worst-case state complexity bounds for the numbers of vertical states can be reached using just binary trees, and for the sake of readability we restrict here consideration to automata operating on ranked trees. The upper bound construction for bottom-up star for unranked tree

automata was given in [20]. The generalized construction relies on the same ideas as Lemma 2 below, however, the notations are considerably more involved.

In the case of DFAs operating on strings, it is common to give state complexity bounds in terms of complete DFAs, that is, all transitions of a DFA are required to be defined, see e.g. [8, 24]. In order to keep our state complexity bounds consistent with corresponding results for tree automata operating on unranked trees [1, 18, 19], our definition allows a deterministic tree automaton to have undefined transitions.

Note that requiring a ranked tree automaton (or an ordinary DFA) to be complete, changes the number of states by at most one. On the other hand, for deterministic tree automata operating on unranked trees where the horizontal languages are defined by DFAs [1, 18, 19], the sizes of an incomplete deterministic automaton and the corresponding completed version may be significantly different. In an unranked tree automaton, adding a dead state q_{sink} for the bottom-up computation, requires the addition, corresponding to an input symbol σ, a horizontal language $L_{\sigma,q_{\text{sink}}}$ that is the complement of a finite disjoint union $L_{\sigma,q_1} \cup \ldots \cup L_{\sigma,q_n}$, where $q_1, \ldots, q_n$ are the vertical states of the incomplete automaton. The size of the minimal DFA for $L_{\sigma,q_{\text{sink}}}$ may be considerably larger than the sum of the sizes of the DFAs for L_{σ,q_i}, $i = 1, \ldots, n$, [9].

2 Basic definitions on tree automata

We assume that the reader is familiar with the basics of automata and formal languages [23, 24]. Here we recall and introduce some definitions related to tree automata. For more information the reader may consult the texts by Gécseg and Steinby [5, 6] or the electronic book by Comon et al. [1].

The cardinality of a finite set S is $|S|$ and the power set of S is 2^S. The set of positive integers is $\mathbb{N}$. A ranked alphabet is a finite set Σ where each element is associated a nonnegative integer as its rank. The set of elements of rank m is Σ_m, $m \geq 0$. The set of trees over ranked alphbet Σ, or Σ-trees, F_Σ, is the smallest set S satisfying the condition: if $m \geq 0$, $\sigma \in \Sigma_m$ and $t_1, \ldots, t_m \in S$ then $\sigma(t_1, \ldots, t_m) \in S$.

A *tree domain* is a prefix-closed subset D of $\mathbb{N}^*$ such that if $ui \in D$, $u \in \mathbb{N}^*$, $i \in \mathbb{N}$ then $uj \in D$ for all $1 \leq j < i$. The set of nodes of a tree $t \in F_\Sigma$ can be represented in the well-known way as a tree domain $\text{dom}(t) \subseteq \{1, \ldots, M\}^*$ where M is the largest rank of any element of the ranked alphabet Σ. The tree t is then viewed as a mapping $t : \text{dom}(t) \to \Sigma$.

We assume that notions such as the *root,* a *leaf,* a *subtree* and the *height* of a tree are known. We use the convention that the height of a single node tree is zero. For $\sigma \in \Sigma$ and $t \in F_\Sigma$, $\text{leaf}(t, \sigma) \subseteq \text{dom}(t)$ denotes the set of leaves of t with label σ. Let t be a tree and u some node of t. The tree obtained from t by replacing the subtree at node u with a tree s is denoted $t(u \leftarrow s)$. The notation is extended in the natural way for a set of pairwise independent nodes U of t and $S \subseteq F_\Sigma$: $t(U \leftarrow S)$ is the set of trees obtained from t by replacing each node of U by some tree in S.

The set of Σ-trees where exactly one leaf is labelled by a special symbol x ($x \notin \Sigma$) is $F_\Sigma[x]$. For $t \in F_\Sigma[x]$ and $t' \in F_\Sigma$, $t(x \leftarrow t')$ denotes the tree obtained from t by replacing the unique occurrence of variable x by t'.

A *deterministic bottom-up tree automaton* (DTA) is a tuple $A = (\Sigma, Q, Q_F, g)$, where Σ is a ranked alphabet, Q is a finite set of states, $Q_F \subseteq Q$ is a set of accepting states and g associates to each $\sigma \in \Sigma_m$ a partial function $\sigma_g : Q^m \longrightarrow Q$, $m \geq 0$. In the usual way, we define the state $t_g \in Q$ reached by A at the root of a tree $t = \sigma(t_1, \ldots, t_m)$, $\sigma \in \Sigma_m$, $m \geq 0$, $t_i \in F_\Sigma$, $i = 1, \ldots, m$, inductively by setting $t_g = \sigma_g((t_1)_g, \ldots, (t_m)_g)$ if the right side is defined, and t_g is undefined otherwise. The tree language recognized by A is $L(A) = \{t \in F_\Sigma \mid t_g \in Q_F\}$. Deterministic bottom-up tree automata recognize the family of regular tree languages.

The intermediate stages of a computation of A, called *configurations of A,* are Σ-trees where some leaves may be labeled by states of A. The set of configurations of A consists of Σ^A-trees where $\Sigma_0^A = \Sigma_0 \cup \{Q\}$ and $\Sigma_m^A = \Sigma_m$ when $m \geq 1$.

A bottom-up automaton begins processing the tree from the leaves because, following a common custom, we view trees to be drawn with the root at the top. As discussed in the previous section, our definition allows a DTA to have undefined transitions, that is, σ_g, $\sigma \in \Sigma_m$, is a partial function.

2.1 Iterated concatenation of trees

We extend the string concatenation operation to an operation where a leaf of a tree is replaced by another tree. Concatenation of trees can be defined also as a parallel operation, however, as will be observed below the iteration of parallel concatenation does not preserve recognizability.

For $\sigma \in \Sigma_0$ and $t_1, t_2 \in F_\Sigma$, we define the *sequential σ-concatenation* of t_1 and t_2 as

$$t_1 \cdot_\sigma^s t_2 = \{\, t_2(u \leftarrow t_1) \mid u \in \mathrm{leaf}(t_2, \sigma) \,\}. \tag{1}$$

That is, $t_1 \cdot_\sigma^s t_2$ is the set of trees obtained from t_2 by replacing one occurrence of a leaf labeled by σ with t_1. The definition is extended in the natural way for tree languages $T_1, T_2 \subseteq F_\Sigma$ by setting

$$T_1 \cdot_\sigma^s T_2 = \bigcup_{t_i \in T_i, i=1,2} t_1 \cdot_\sigma^s t_2.$$

Alternatively, we can consider a *parallel σ-concatenation* of tree languages $T_1, T_2 \subseteq F_\Sigma$ by setting

$$T_1 \cdot_\sigma^p T_2 = \{\, t_2(\mathrm{leaf}(t_2, \sigma) \leftarrow T_1) \mid t_2 \in T_2 \,\}.$$

The operation $T_1 \cdot_\sigma^p T_2$ is called the σ-product of T_1 and T_2 in [6]. Note that the parallel concatenation of tree languages could not be defined by defining first the concatenation of individual trees (as was done for sequential concatenation in (1)) and then taking union over sets of trees. For trees $t_1, t_2 \in F_\Sigma$, $t_1 \cdot_\sigma^p t_2$ is an individual tree while $t_1 \cdot_\sigma^s t_2$ is a set of trees. In the case where no leaf of t_2 is labeled by σ, $t_1 \cdot_\sigma^s t_2 = \emptyset$ and $t_1 \cdot_\sigma^p t_2 = t_2$.

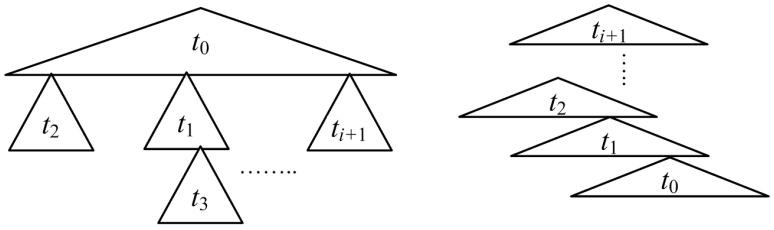

Figure 1: A tree in $T_\sigma^{s,t,*}$ (a) and in $T_\sigma^{s,b,*}$ (b). Here $t_0, t_1, \ldots t_{i+1}$ are trees in T.

When considering bottom-up tree automata operating on unary trees, both of the above definitions reduce to the usual concatenation of string languages: when processing $T_1 \circ T_2$, $\circ \in \{\cdot_\sigma^s, \cdot_\sigma^p\}$, the automaton reads first an element of T_1 and then an element of T_2.

The parallel concatenation operation is associative, however, sequential concatenation is nonassociative, as observed below in Example 1. The nonassociativity of sequential concatenation means, in particular, that there are two variants of the iteration of the operation.

For $\sigma \in \Sigma$ and $T \subseteq F_\Sigma$, we define the *kth sequential top-down σ-power of T,* $k \geq 0$, by setting $T_\sigma^{s,t,0} = \{\sigma\}$, and $T_\sigma^{s,t,k} = T \cdot_\sigma^s T_\sigma^{s,t,k-1}$, when $k \geq 1$. The *sequential top-down σ-star of T* is then

$$T_\sigma^{s,t,*} = \bigcup_{k \geq 0} T_\sigma^{s,t,k}.$$

Similarly, the *kth sequential bottom-up σ-power of T,* is defined by setting $T_\sigma^{s,b,0} = \{\sigma\}$, $T_\sigma^{s,b,1} = T$ and $T_\sigma^{s,b,k} = T_\sigma^{s,b,k-1} \cdot_\sigma^s T$, when $k \geq 2$. The *sequential bottom-up σ-star of T* is

$$T_\sigma^{s,b,*} = \bigcup_{k \geq 0} T_\sigma^{s,b,k}.$$

Note that the definition of bottom-up σ-powers explicitly sets $T_\sigma^{s,b,1}$ to be equal to T. This is done because $T_\sigma^{s,b,0} \cdot_\sigma^s T$ can be a strict subset of T if some trees of T contain no occurrences of σ. Figure 1 illustrates the definitions of top-down star and bottom-up star.

Example 1. It is easy to see that sequential concatenation is non-associative. Consider a ranked alphabet Σ determined by $\Sigma_2 = \{\omega\}$, $\Sigma_0 = \{\sigma\}$ and let $t = \omega(\sigma, \sigma)$. Now $t \cdot_\sigma^s t = \{\omega(\omega(\sigma,\sigma),\sigma), \omega(\sigma, \omega(\sigma,\sigma))\}$ and $t_1 = \omega(\omega(\sigma,\sigma), \omega(\sigma,\sigma)) \in t \cdot_\sigma^s (t \cdot_\sigma^s t)$ but, on the other hand, $t_1 \notin (t \cdot_\sigma^s t) \cdot_\sigma^s t$.

To illustrate the difference of top-down and bottom-up star, respectively, consider $T = \{\omega(\sigma, \sigma)\}$. We note that $T_\sigma^{s,t,*} = F_\Sigma$ and

$$T_\sigma^{s,b,*} = \{r \in F_\Sigma \mid \text{ each non-leaf node of } r \text{ has at least one leaf as a child } \}.$$

Note that with $T = \{\omega(\sigma, \sigma)\}$, $T_\sigma^{s,b,k}$, $k \geq 0$, consists of trees of height (exactly) k. The trees of $T_\sigma^{s,b,*}$ all consist of a path labeled by binary symbols ω and all children of nodes of the path that "diverge" from the path are labeled by the leaf symbol σ.

The following characterization of bottom-up σ-star as the smallest set closed under concatenation with T from the right follows directly from the definition of bottom-up star. The characterization will be used in the next section.

Lemma 1. *For $\sigma \in \Sigma_0$ and $T \subseteq F_\Sigma$, define $\mathrm{cl}_\sigma(T)$ as the smallest set $S \subseteq F_\Sigma$ such that (i) $T \cup \{\sigma\} \subseteq S$, and (ii) $t_1 \cdot_\sigma^s t_2 \in S$ for every $t_2 \in T$ and $t_1 \in S$. Then $\mathrm{cl}_\sigma(T) = T^{s,b,*}$.*

Completely analogously we can define, for $T \subseteq F_\Sigma$, the parallel σ-star of T, denoted $T_\sigma^{p,*}$. Since parallel concatenation is associative, we do not need to distinguish the bottom-up and top-down variants. However, we note that with $T = \{\omega(\sigma, \sigma)\}$, $T_\sigma^{p,*}$ consists of all balanced trees over the ranked alphabet Σ, where $\Sigma_2 = \{\omega\}$, $\Sigma_0 = \{\sigma\}$. Since the "straightforward" definition of Kleene-star based on parallel concatenation does not preserve regularity, in fact, Gécseg and Steinby [6] define a regularity preserving σ-iteration operation by defining the kth ($k \geq 1$) power of T by parallel-concatenating the union of all the ith powers of T, $0 \leq i \leq k-1$, with the tree language T.

It is easy to verify that the definition of the σ-iteration operation (based on parallel concatenation) given in section 7 of [6] coincides with the sequential top-down star defined above, and in the following we will focus only on the sequential variants of iterated concatenation. The top-down (respectively, bottom-up) σ-powers and σ-star of a tree language T are in the following denoted $T_\sigma^{t,k}$, ($k \geq 0$), and $T_\sigma^{t,*}$ (respectively, $T_\sigma^{b,k}$ and $T_\sigma^{b,*}$), that is, we drop the superscript "s" in the notation.

3 Bottom-up and top-down star: state complexity

We establish for the bottom-up star operation a tight state complexity bound that is of a different order of magnitude than the state complexity of Kleene-star for string languages. First we give an upper bound for the state complexity of bottom-up star.

Lemma 2. *Suppose that tree language L is recognized by a DTA with n states. For $\sigma \in \Sigma_0$, the tree language $L_\sigma^{b,*}$ can be recognized by a DTA with $(n + \frac{3}{2})2^{n-1}$ states.*

Proof. Let $A = (\Sigma, Q, Q_F, g_A)$ be a DTA with n states recognizing the tree language L. Without loss of generality we assume that σ_{g_A} is defined, because otherwise

$$L(A)_\sigma^{b,*} = L(A)_\sigma^{b,0} \cup L(A)_\sigma^{b,1} = \{\sigma\} \cup L(A),$$

and it is easy to construct a DTA with $n+1$ states that recognizes $L(A) \cup \{\sigma\}$.

Choose three disjoint subsets of $2^Q \times (Q \cup \{\mathrm{dead}\})$ by setting

(i) $P_1 = \{(S, q) \mid S \in 2^Q, \{q, \sigma_{g_A}\} \subseteq S, q \in Q_F\}$,

(ii) $P_2 = \{(S, q) \mid S \in 2^Q, q \in S \cap (Q - Q_F)\}$,

(iii) $P_3 = \{(S, \text{dead}) \mid S \in 2^Q, S \neq \emptyset\}$.

Here dead is a new element not in Q. Now define a DTA $B = (\Sigma, P, P_F, g_B)$ where

$$P = P_1 \cup P_2 \cup P_3 \cup \{p_{\text{new}}\}, \quad P_F = \{(S, q) \in P \mid S \cap Q_F \neq \emptyset\} \cup \{p_{\text{new}}\}.$$

We define the transitions of B by setting, $\sigma_{g_B} = p_{\text{new}}$, and for $\tau \in \Sigma_0 - \{\sigma\}$,

$$\tau_{g_B} = \begin{cases} (\{\tau_{g_A}, \sigma_{g_A}\}, \tau_{g_A}) & \text{if } \tau_{g_A} \in Q_F, \\ (\{\tau_{g_A}\}, \tau_{g_A}) & \text{if } \tau_{g_A} \in Q - Q_F, \\ \text{undefined}, & \text{if } \tau_{g_A} \text{ is undefined.} \end{cases} \tag{2}$$

To define transitions on Σ_m, $m \geq 1$, we view p_{new} as the state $(\{\sigma_{g_A}\}, \sigma_{g_A})$, and hence every state of B is represented in the form (S, q), $S \subseteq Q$, $q \in Q$. (Note that p_{new} is not the same as $(\{\sigma_{g_A}\}, \sigma_{g_A})$, because the former is an accepting state and the latter need not be accepting.) For $\tau \in \Sigma_m$ and $(S_1, q_1), \ldots, (S_m, q_m) \in P$, we first denote

$$X = \bigcup_{i=1}^{m} \{\tau_{g_A}(q_1, \ldots, q_{i-1}, z, q_{i+1}, \ldots, q_m) \mid z \in S_i\}$$

Now we define

$$\tau_{g_B}((S_1, q_1), \ldots, (S_m, q_m)) \tag{3}$$

to be equal to

(i) $(X \cup \{\sigma_{g_A}\}, \tau_{g_A}(q_1, \ldots, q_m))$ if $\tau_{g_A}(q_1, \ldots, q_m) \in Q_F$,

(ii) $(X, \tau_{g_A}(q_1, \ldots, q_m))$ if $\tau_{g_A}(q_1, \ldots, q_m) \in Q - Q_F$,

(iii) (X, dead) if $X \neq \emptyset$ and $\tau_{g_A}(q_1, \ldots, q_m)$ is undefined.

In the remaining case, where $X = \emptyset$ and $\tau_{g_A}(q_1, \ldots, q_m)$ is undefined, also (3) is undefined. Note that if for some $1 \leq i \leq m$, $q_i = \text{dead}$, this implies automatically that $\tau_{g_A}(q_1, \ldots, q_m)$ is undefined.

Recall that if (S, q), $S \subseteq Q$, $q \in Q$ is a state of B then $q \in S$ and, furthermore, if $q \in Q_F$ then $\sigma_{g_A} \in S$. The transitions of g_B preserve this property and the state in (i) (in (ii), (iii), respectively) is an element of P_1 (an element of P_2, P_3, respectively).

The second component of the state of B simply simulates the computation of A on the current subtree, and goes to the state dead if the next state of A is undefined. Intuitively, the first component of the state of B consists of all states that A could reach at the current subtree t' assuming that

$$\text{in } t' \text{ at most one subtree of } L(A)^{b,k}_{\sigma}, \ k \geq 0, \text{ has been replaced by a leaf } \sigma. \tag{4}$$

Inductively, assume that B assigns to the root of tree t_i a state $(S_i, (t_i)_{g_A})$ where $S_i \subseteq Q$ satisfies the property (4) for t_i, $i = 1, \ldots, m$. Now the rule (3) assigns to the

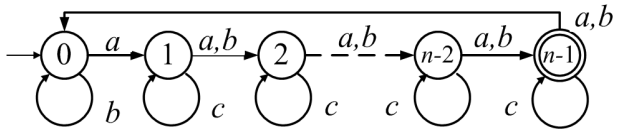

Figure 2: The DFA A from [24] with added c-transitions.

root of tree $t = \tau(t_1, \ldots, t_m)$ a state (S, q) where $q = \tau_{g_A}((t_1)_{g_A}, \ldots, (t_m)_{g_A})$ and S consists of all states that A could reach at the root of t assuming the computation uses as arguments $q_1, \ldots, q_m$ where (by the definition of the set X) at most one of the q_i's can be replaced by an arbitrary state from S_i, $1 \leq i \leq m$. This means that the state (S, q) again satisfies the property (4) for the tree t.

The choice of the set of final states P_F and Lemma 1 now imply that $L(B) = L(A)^{b,*}_{\sigma}$.

It remains to estimate the worst-case size of B. We note that if $Q_F = \{\sigma_{g_A}\}$, in B only states of the form $(\{q\}, q)$, $q \in Q$, can be reachable, and p_{new} can be identified with $(\{\sigma_{g_A}\}, \sigma_{g_A})$. In this case $L(A)^{b,*}_{\sigma}$ has a DTA with n states. Thus, without loss of generality we assume that Q_F contains a final state distinct from σ_{g_A}.

We note that $|P_1| = |Q_F| \cdot 2^{n-2}$, $|P_2| = |Q - Q_F| \cdot 2^{n-1}$ and $|P_3| = 2^n - 1$. Here the estimation of the size of P_1 relies on the above observation that we can exclude the possibility $Q_F = \{\sigma_{g_A}\}$. Thus, the cardinality of $P_1 \cup P_2 \cup P_3 \cup \{p_{\text{new}}\}$ is maximized as $(n + \frac{3}{2})2^{n-1}$ when $|Q_F| = 1$. □

The upper bound of Lemma 2 is of a different order of magnitude than the known state complexity of Kleene-star for string languages [24]. It remains to verify that the bound of Lemma 2 can be reached in the worst case.

Figure 2 represents a DFA A used in [24, 25] for the lower bound construction for Kleene-star where we have added transitions on the symbol c. Note that A is an incomplete DFA since the c transition on 0 is undefined. Based on A we define in the following a tree automaton M_A.

Choose $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$ where $\Sigma_0 = \{e\}$, $\Sigma_1 = \{a, b, c\}$ and $\Sigma_2 = \{a_2, d_2\}$. We define a DTA $M_A = (\Sigma, Q_A, Q_{A,F}, g_A)$, where $Q_A = \{0, 1, \ldots, n-1\}$, $Q_{A,F} = \{n-1\}$ and the transition function g_A is defined by setting:

(i) $e_{g_A} = 0$, $c_{g_A}(i) = i$, $1 \leq i \leq n-1$,

(ii) $a_{g_A}(i) = (a_2)_{g_A}(i, i) = i + 1$, $0 \leq i \leq n-2$,
$a_{g_A}(n-1) = (a_2)_{g_A}(n-1, n-1) = 0$,

(iii) $b_{g_A}(i) = i + 1$, $1 \leq i \leq n-2$, $b_{g_A}(j) = 0$, $j \in \{0, n-1\}$,

(iv) $(d_2)_{g_A}(0, i) = i$, $i = 0, 2, 3, \ldots, n-1$, $(d_2)_{g_A}(1, 1) = 1$.

All transitions of g_A not listed above are undefined. Intuitively, the construction of M_A can be, roughly speaking, explained as follows. Denote by T_d the subset of

F_Σ consisting of trees without any occurrences of the binary symbol d_2, thus the only binary symbol in trees of T_d is a_2. On a tree $t \in T_d$, the DTA M_A simulates the computation of A on each string of symbols starting from a node of height one, where occurrences of a_2 are "interpreted" simply as a. The computations on different paths verify that for any $u \in \mathrm{dom}(t)$ labeled by a_2 and any nodes v_1 and v_2 of height one below u, the simulated computations started from v_1 and v_2 agree at u.

Note that the original DFA has no transitions on d, and the transitions on d_2 have been added for a technical reason that will be used in the proof of Lemma 4. Also, the above intuitive description is not completely precise on how M_A operates on binary symbols a_2 where one child is a leaf (that gets assigned the state 0) and the other child is not a leaf. The following Lemmas 3 and 4 rely only on the formal definition of the transition function g_A of M_A. The above intuitive description of the operation of M_A is intended only as a guide that may be useful in understanding the operation of the DTA constructed to recognize the bottom-up e-star of $L(M_A)$. Finally, note that the d_2-transitions will be needed only to establish the reachability of one particular state, and in most of the technical constructions the above intuitive description of the operation of M_A (based on the DFA A of Figure 2) can be sufficient.

Using the construction of the proof of Lemma 2, based on M_A we construct a DTA $M_B = (\Sigma, Q_B, Q_{B,F}, g_B)$ that recognizes the tree language $L(M_A)_e^{b,*}$. We make the convention that the sink-state "dead" used in the proof is denoted by n. Thus the set of states Q_B consists of the special state p_{new} assigned to e and all pairs

$$(P, q),\ P \subseteq \{0, \ldots, n-1\},\ 0 \leq q \leq n, \tag{5}$$

where $0 \leq q \leq n-1$ implies $q \in P$, $q = n-1$ implies $0 \in P$ and $q = n$ implies $P \neq \emptyset$. The number of pairs as in (5) is $(n + \frac{3}{2})2^{n-1} - 1$.

In the following two lemmas we establish that M_B is a minimal DTA. That is, first we show that all states of Q_B are pairwise inequivalent with respect to the Myhill-Nerode equivalence relation extended to trees. Second we show that all states of Q_B are reachable, that is, for each $q \in Q_B$ there exists $t \in F_\Sigma$ such that $t_{g_B} = q$. The proof of our first lemma assumes that all states are reachable which will be established next in Lemma 4[1].

Lemma 3. *All states of M_B are pairwise inequivalent.*

Proof. For the sake of convenience, we assume that we have already proven that all states of M_B are reachable (Lemma 4). Thus, in order to distinguish two states with respect to the Myhill-Nerode relation, we can use an arbitrary configuration of M_B where one leaf is replaced by the given states. More formally, in order to show that two distinct states of Q_B, p_1 and p_2, are inequivalent, it is sufficient to find $t \in F_{\Sigma^{M_B}}[x]$ such that the computation of M_B started from the configuration $t(x \leftarrow p_1)$ accepts if and only if the computation started from the configuration $t(x \leftarrow p_2)$ does not accept.

[1]The proof of Lemma 4 does not rely on Lemma 3.

We first show that any two distinct states (S_1, q_1) and (S_2, q_2) as in (5) are not equivalent. After that we consider the special state p_{new}. We begin by considering the case where neither of q_1 or q_2 is equal to n (which was used to denote the dead state of M_A).

Case $0 \leq q_1, q_2 \leq n-1$: (a) Assume $S_1 \neq S_2$ and $s \in S_1 - S_2$ (The other possibility is completely symmetric.) After reading $n-s-1$ unary symbols a, a final state is reached from state (S_1, q_1). On the other hand, since (S_2, q_2) is as in (5), $q_2 \neq s$. This means that the computation C that begins with (S_2, q_2) and reads $n-s-1$ unary symbols a ends with a non-final state. Note that at some point during the computation C, the second component may become $n-1$ which adds an element 0 to the first component. However, at the end of the computation C the first component cannot contain $n-1$.

(b)(i) Next we consider the case $S_1 = S_2 = S$, $\{0, 1, \ldots, n-2\} \not\subseteq S$ and $q_1 \neq q_2$. According to the definition of the states (5), $q_1, q_2 \in S$. Choose $p \in \{0, 1, \ldots, n-2\} - S$ and consider a tree $t_1 = a^{2n-2-q_1} a_2((\{q_1, p\}, p), x) \in F_{\Sigma^{M_B}}[x]$. Since $p \in \{0, 1, \ldots, n-2\}$, $(\{q_1, p\}, p)$ is a legal state (5). Consider the computation of M_B on tree $t_1(x \leftarrow (S, q_1))$. Since $p \notin S$ the state $(\{q_1 + 1\}, n)$ is assigned to the root of the subtree $a_2((\{q_1, p\}, q_1), (S, q_1))$. (Here addition is modulo n.) After this the computation reads the $2n-2-q_1$ unary symbols a in t_1 and ends in an accepting state. On the other hand, consider the computation of M_B on $t_1(x \leftarrow (S, q_2))$. Since $p \notin S$ and $q_2 \notin \{q_1, p\}$, the transition $(a_2)_{g_B}$ on arguments $(\{q_1, p\}, p)$, $(S, q_2))$ is undefined and the computation does not accept.

(b)(ii) Consider $S = \{0, 1, \ldots, n-2\}$, and hence we know that $q_1, q_2 \neq n-1$. From state (S, q_i) by reading a unary symbol b we get (S', q_i'), where $S' = \{0, 2, \ldots, n-2, n-1\}$. Since $q_1, q_2 \neq n-1$, $q_1' \neq q_2'$ and the states (S', q_1') and (S', q_2') are distinguished as in b(i) above.

(b)(iii) Consider then the possibility $S = \{0, 1, \ldots, n-1\}$ and $q_1 \neq q_2$. If $\{q_1, q_2\} \neq \{0, n-1\}$, by reading a unary symbol b from (S, q_1) and (S, q_2), respectively, we get two states (S', q_1'), (S', q_2'), $q_1' \neq q_2'$, that are distinguished as in the previous case[2]. Next consider the case $\{q_1, q_2\} = \{0, n-1\}$, and first assume that $n \geq 3$. By reading a unary symbol a we obtain states $(S, q_1 + 1)$, $(S, q_2 + 1)$ where $q_1 + 1 \neq q_2 + 1$ and $q_i + 1 \neq n-1$, $i = 1, 2$ (addition is modulo n). The states $(S, q_1 + 1)$ and $(S, q_2 + 1)$ can be distinguished as in the previous cases.

Finally consider the possibility $n = 2$ and $\{q_1, q_2\} = \{0, 1\}$. From state $(\{0, 1\}, 1)$ by reading unary symbols ca, we reach the accepting state $(\{0, 1\}, 0)$. On the other hand, a computation starting from $(\{0, 1\}, 0)$ by reading the unary symbols ca reaches the nonaccepting state $(\{0\}, 2)$.

Case where $q_2 = n$: First assume $q_1 \neq n$. Choose $t_2 \in F_{\Sigma^{M_B}}[x]$ by setting $t_2 = a^{n-2} a_2((\{0, 1\}, 1), b^{n-1}(x))$. Since $n-1$ consecutive b-transitions take any

[2]The b-transitions of A violate injectivity only on states 0 and $n-1$.

state of A to state 0, the computation of M_B on $t_2(x \leftarrow (S_1, q_1))$ assigns state $(\{0\}, 0)$ to the root of the subtree $b^{n-1}((S_1, q_1))$. Then the state $(\{1\}, n)$ is reached at the root of the subtree $a_2((\{0,1\}, 1), b^{n-1}((S_1, q_1)))$. A final state $(\{n-1\}, n)$ is reached after reading further $n-2$ unary symbols a. On the other hand, in the computation of M_B on $t_2(x \leftarrow (S_2, n))$ the state $(\{0\}, n)$ is assigned to the root of the subtree $b^{n-1}((S_2, n))$. When reading the binary symbol a_2 with arguments $(\{0,1\}, 1)$ and $(\{0\}, n)$ the computation step of M_B is undefined, and hence M_B does not accept $t_2(x \leftarrow (S_2, n))$.

Finally consider the case where also $q_1 = n$. Thus $S_1 \neq S_2$ and choose $s \in S_1 - S_2$. After reading $n - s - 1$ unary symbols a, a final state is reached from state (S_1, n), and the same computation does not reach a final state from (S_2, n).

It remains to show that p_{new} is not equivalent with any state (S, q) as in (5). Since p_{new} is final, it is sufficient to consider states where $n-1 \in S$. Thus, by reading a unary symbol c from state (S, q) we get a state (S', q'), where $n-1 \in S'$ and $0 \leq q' \leq n$. On the other hand, computations starting from p_{new} are identical to computations starting from $(\{0\}, 0)$ and hence a computation step with unary symbol c is undefined. □

Before the next lemma we introduce the following notation. For a unary tree representing a configuration of M_B, $t = z_1(z_2(\dots z_m(z_0)\dots)) \in F_{\Sigma^{M_B}}$, we define $\text{word}(t) = z_m z_{m-1} \dots z_1$. Note that $\text{word}(t)$ consists of the sequence of symbols labeling the nodes of t bottom-up, and the label of the leaf is not included. In the following when we refer to $\text{word}(t)$ of a tree t, without further mention, this implies that t is a unary tree.

Lemma 4. *All states of M_B are reachable.*

Proof. The transition function of M_B assigns the special state p_{new} to leaf symbol e. Recall that from p_{new} the computation of M_B continues as from $(\{0\}, 0)$. Thus, after reading $n-1$ unary symbols a we reach the state $(\{0, n-1\}, n-1)$.

Inductively, we assume that a state $(\{0, 1, 2, \dots, k, n-1\}, n-1)$, $0 \leq k < n-2$, is reachable. We show that $(\{0, 1, 2, \dots, k+1, n-1\}, n-1)$ is also reachable. From state $(\{0, 1, 2, \dots, k, n-1\}, n-1)$, we reach the state $Z_1 = (\{1, 2, \dots, k+1, 0\}, 0)$ by reading a unary symbol a. By our assumption on k, $k+1 < n-1$. Thus from Z_1 we reach the state $Z_2 = (\{2, 3, \dots, k+2, 0\}, 0)$ by reading b. Since $k < n-2$, all elements of $\{2, 3, \dots, k+2, 0\}$ are distinct (that is, the b-transition does not take $k+1$ to 0). After reading $n-1$ symbols a, the state $(\{1, 2, \dots, k+1, n-1, 0\}, n-1)$ is reached. The element 0 is added to the first component as the second component becomes $n-1$.

By the above inductive claim we now know that the state $(\{0, 1, \dots, n-2, n-1\}, n-1)$ is reachable. After reading $i+1$ $a's$, state $(\{0, 1, \dots, n-2, n-1\}, i)$ is reached, $0 \leq i \leq n-1$.

Inductively, assume that all states (S, j), where $|S| \geq k+1$, $1 \leq k < n$ and $0 \leq j \leq n-1$ as in (5) are reachable. We show that then also states where

$|S| = k$ are reachable. Let (S, s_i) where $S = \{s_1, s_2, \ldots, s_k\}$, $1 \leq i \leq k$ and $0 \leq s_1 < s_2 < \ldots < s_k \leq n-1$ be an arbitrary state where $|S| = k$. Recall that in states of M_B, when the second component is not n, it must belong to the first component.

In the below cases (a) and (b), numbers $z \geq n$ are interpreted as the unique element of $\{0, 1, \ldots, n-1\}$ congruent to z modulo n.

(a-i) First consider the case where $s_i < n-1$. The following discussion assumes $n \geq 3$, and the case $n = 2$ is handled in case (a-ii). Since $|S| = k < n$, in the "cyclical sequence" of $s_1, \ldots, s_k$, there exist two consecutive numbers with difference at least two, where the difference between the numbers s_k and s_1 is counted modulo n. More formally, either there exists $1 \leq j \leq k-1$ such that $s_{j+1} - s_j \geq 2$ or $n + s_1 - s_k \geq 2$. In the latter case we choose $j = k$. In the following we assume that $i \leq j$. The case where $i > j$ is similar and only some notations are changed. According to the inductive assumption, the state $Z_3 = (\{0, n-1\} \cup S_1, n + s_i - s_j - 1)$ where $S_1 = \{s_{j+1} - s_j - 1, s_{j+2} - s_j - 1, \ldots, s_k - s_j - 1, n + s_1 - s_j - 1, n + s_2 - s_j - 1, \ldots, n + s_{j-1} - s_j - 1\}$ is reachable. Note that since $0 \leq s_1 < s_2 < \ldots < s_k \leq n-1$ and $s_{j+1} - s_j \geq 2$, $|S_1 \cup \{0, n-1\}| = k+1$. After reading from state Z_3 a unary symbol b, we get the state $Z_4 = (\{0\} \cup S_2, n + s_i - s_j)$ where $S_2 = \{s_{j+1} - s_j, s_{j+2} - s_j, \ldots, s_k - s_j, n + s_1 - s_j, n + s_2 - s_j, \ldots, n + s_{j-1} - s_j\}$. Since $0 \leq s_1 < s_2 < \ldots < s_k \leq n-1$, $0 \notin S_2$. From state Z_4 we reach the state $(\{s_j, s_{j+1}, s_{j+2}, \ldots, s_k, n + s_1, n + s_2, \ldots, n + s_{j-1}\}, n + s_i)$ by reading s_j symbols a. The latter state is the state (S, s_i) that we wanted.

(a-ii) Assume that $s_i < n-1$ and $n = 2$. Now $k = 1$, and the only legal state (S, s_i), $|S| = k = 1$, $0 \leq s_i < 1$, is $(\{0\}, 0)$ (because we know that $s_i \in S$). The state $(\{0\}, 0)$ is reached from state p_{new} by reading unary symbols ab.

(b) Now consider the case where $s_i = n-1$, and thus $i = k$. This implies that $0 \in S$, and we have $s_i (= s_k) = n-1$ and $s_1 = 0$. Since $k < n$, there exists $1 \leq j \leq k-1$ such that $s_{j+1} - s_j \geq 2$. According to the inductive assumption, the state $Z_5 = (\{0, n-1\} \cup S_3, n - 2 - s_j)$ is reachable, where $S_3 = \{s_{j+1} - s_j - 1, s_{j+2} - s_j - 1, \ldots, s_{k-1} - s_j - 1, n - 1 - s_j - 1, n + 0 - s_j - 1, n + s_2 - s_j - 1, \ldots, n + s_{j-1} - s_j - 1\}$. Similarly as in (a) above we observe that $|S_3 \cup \{0, n-1\}| = k+1$. From state Z_5 we get the state $Z_6 = (\{s_{j+1} - s_j, s_{j+2} - s_j, \ldots, s_{k-1} - s_j, n - 1 - s_j, n + 0 - s_j, n + s_2 - s_j, \ldots, n + s_{j-1} - s_j, 0\}, n - 1 - s_j)$ by reading a symbol b. After reading s_j symbols a, from state Z_6 we reach the state $(\{s_{j+1}, s_{j+2}, \ldots, s_{k-1}, n-1, n + 0, n + s_2, \ldots, n + s_{j-1}, s_j\}, n-1)$. This means that we have reached the desired state $(S, n-1)$ with $S = \{0, s_2, \ldots, s_{k-1}, n-1\}$.

Up to now, we have shown that all that states (S, j), $S \subseteq \{0, \ldots, n-1\}$, $0 \leq j \leq n-1$ as in (5) are reachable. Next we will show that the states (S, n), $S \subset \{0, 1, \ldots, n-1\}$ are reachable.

We know that $(\{0, 1, \ldots, n-1\}, 0)$ is reachable and from this state we get $Z_7 = (\{1, \ldots, n-1\}, n)$ by reading a unary symbol c. From Z_7 we get all states

(S, n), $|S| = n-1$ by cycling the elements of S using a-transitions. Now inductively, assume that all states (S, n), $n > |S| \geq k+1$, $k < n-1$ are reachable. Consider an arbitrary state (S, n) where $|S| = k$. Choose $0 \leq j \leq n-1$ such that $j \notin S$. By our inductive assumption the state $(S \cup \{j\}, n)$ is reachable. From this state we reach (S, n) by reading the sequence of unary symbols $a^{n-j}ca^j$. Note that transitions on a always add one modulo n to states of S and the c-transition deletes the element 0 and is the identity on all other elements.

It remains to consider the state $(\{0, 1, \ldots, n-1\}, n)$. We know that states $(\{0, 1\}, 0)$ and $(\{0, 1, \ldots, n-1\}, 1)$ are reachable. According to the definition of d_2-transitions of M_A, the d_2-transition of M_B with arguments $(\{0, 1\}, 0)$ and $(\{0, 1, \ldots, n-1\}, 1)$ gives the state $(\{0, 1, \ldots, n-1\}, n)$. □

Note that above the transitions on d_2 were needed only to establish that the state $(\{0, 1, \ldots, n-1\}, n)$ is reachable in M_B. The transitions of d_2 in M_A did not have a similar intuitive interpretation as the other transitions based on the DFA A, and they were introduced only for the technical purpose needed at the end of the proof of Lemma 4.

By Lemmas 2, 3 and 4 we have a tight bound for the state complexity of bottom-up star that differs by an order of magnitude from the known bound for Kleene-star of string languages [4, 24].

Theorem 1. *If A is a DTA with n states, the bottom-up star of $L(A)$ can be recognized by a DTA with $(n + \frac{3}{2}) \cdot 2^{n-1}$ states. For every $n \geq 2$, there exists an n-state DTA A and $\sigma \in \Sigma_0$ such that the minimal DTA for $L(A)^{b,*}_\sigma$ has $(n + \frac{3}{2}) \cdot 2^{n-1}$ states.*

Next we give a tight state complexity bound for top-down star of regular tree languages. The top-down iteration of the concatenation operation allows the replacement of subtrees at arbitrary locations and, as can perhaps be expected, the state complexity is similar as for the Kleene-star of string languages. However, it should be noted that we are considering incomplete automata and the known state complexity bounds for ordinary DFAs are stated in terms of complete DFAs [24, 25]. The state complexity results for complete and incomplete DFAs, respectively, differ slightly for operations such as union or concatenation [24, 18].

Theorem 2. *Let $A = (\Sigma, Q_A, Q_{A,F}, g_A)$ be a DTA with n states and $\sigma \in \Sigma_0$. The top-down σ-star of the tree language recognized by A, $L(A)^{t,*}_\sigma$, can be recognized by a DTA B with $\frac{3}{4} \cdot 2^n$ states and this bound can be reached in the worst case.*

Proof. The construction of $B = (\Sigma, Q_B, Q_{B,F}, g_B)$ is similar as the construction used to recognize the Kleene-star of a string language. The set of states Q_B consists of nonempty subsets of $P \subseteq Q_A$ such that $P \cap Q_{A,F} \neq \emptyset$ implies $\sigma_{g_A} \in P$, and additionally Q_B has one new state q_{new} that is reached at leaves labeled by σ (the symbol that defines the star operation). Note that the state q_{new} is used as a copy of σ_{g_A} because the latter state is not, in general, accepting. The cardinality of Q_B is maximized as $2^n - 1 - 2^{n-2} + 1 = \frac{3}{4} \cdot 2^n$ by choosing $|Q_{A,F}| = 1$. We leave details of the construction to the reader.

When restricted to unary trees, the top-down (or bottom) star operation coincides with Kleene-star on string languages. Theorem 5.5 of [24] gives a complete DFA C with n states such that the state complexity of the Kleene-star of $L(C)$ is $\frac{3}{4} \cdot 2^n$. Furthermore, C does not have a dead state, which means that the same lower bound construction works for incomplete DFAs. □

4 Kleene-Star Combined with Concatenation

The worst case state complexity of star–of–concatenation of string languages is known [2]. However, already in the case of string languages determining the precise state complexity of combined operations is often quite involved [2, 10].

For tree languages we consider a restricted case of Kleene-star combined with concatenation where one of the arguments for concatenation is the set of all trees F_Σ. For some of the combined operations we get tight bounds that are significantly lower than the function composition of the state complexity of the individual operations. Altogether there are four combinations of bottom-up star (or top-down star) with the parallel or sequential concatenation with the set of all trees. The combined operations for bottom-up star are as follows:

$$(F_\Sigma \cdot^p_\sigma L)^{b,*}_\sigma, \quad (L \cdot^p_\sigma F_\Sigma)^{b,*}_\sigma, \quad (L \cdot^s_\sigma F_\Sigma)^{b,*}_\sigma, \text{ and } (F_\Sigma \cdot^s_\sigma L)^{b,*}_\sigma.$$

It turns out that, for the first and the last of the listed combined operations, the tree automaton constructions can be significantly simplified by relying on general observations about the (parallel or sequential) concatenation of a general tree language with the set of all trees.

Lemma 5. *Let $L \subseteq F_\Sigma$ and $\sigma \in \Sigma_0$. Then*

(i) $(F_\Sigma \cdot^p_\sigma L)^{b,*}_\sigma = (F_\Sigma \cdot^p_\sigma L)^{t,*}_\sigma = F_\Sigma \cdot^p_\sigma L \cup \{\sigma\}$,

(ii) $(L \cdot^s_\sigma F_\Sigma)^{b,*}_\sigma = (L \cdot^s_\sigma F_\Sigma)^{t,*}_\sigma = L \cdot^s_\sigma F_\Sigma \cup \{\sigma\}$.

Using Lemma 5, we get tight state complexity bounds for two combined operations involving bottom-up star and top-down star, respectively.

Theorem 3. *Let A be a DTA with n states and $\sigma \in \Sigma_0$. Then, $(F_\Sigma \cdot^p_\sigma L(A))^{b,*}_\sigma$ can be recognized by a DTA with $2^{n-1} + 1$ states and this bound can be reached in the worst case.*

Proof. Let $A = (\Sigma, Q_A, Q_{A,F}, g_A)$ be a DTA with n states recognizing the tree language L. Without loss of generality we assume that σ_{g_A} is defined, because otherwise $F_\Sigma \cdot^p_\sigma L(A) = L(A)$, and $(F_\Sigma \cdot^p_\sigma L(A))^{b,*}_\sigma = L(A) \cup \{\sigma\}$ and we can easily construct a DTA with $n + 1$ states to recognize $L(A) \cup \{\sigma\}$.

We define a DTA $B = (\Sigma, Q_B, Q_{B,F}, g_B)$ where

$$Q_B = 2^{Q_A} \cup \{q_{\text{new}}\}, \quad Q_{B,F} = \{P \in Q_B \mid P \cap Q_{A,F} \neq \emptyset\} \cup \{q_{\text{new}}\},$$

and the transitions of g_B are defined as below. Note that q_{new} can be viewed as a copy of the state $\{\sigma_{g_A}\}$. The reason why we have an additional state q_{new} is because q_{new} needs to be an accepting state and $\{\sigma_{g_A}\}$ is not accepting, in general.

For $\tau \in \Sigma_0$, $\tau \neq \sigma$, $\tau_{g_B} = \{\tau_{g_A}, \sigma_{g_A}\}$, and, $\sigma_{g_B} = q_{\text{new}}$. For $P \in Q_B$, define $\overline{P} \subseteq Q_A$ by

$$\overline{P} = \begin{cases} P \text{ if } P \in 2^{Q_A}, \\ \{\sigma_{g_A}\} \text{ if } P = q_{\text{new}}. \end{cases}$$

Now for $\tau \in \Sigma_k, k \geq 1$, and $P_i \in Q_B, i = 1, \ldots, k$, define

$$\tau_{g_B}(P_1, \ldots, P_k) = \tau_{g_A}(P_1, \ldots, P_k) \cup \{\sigma_{g_A}\}.$$

We leave to the reader the details of verifying that B recognises the tree language $F_\Sigma \cdot^p_\sigma L(A) \cup \{\sigma\}$. Among the states $P \in Q_B$, the sets where $\sigma_{g_A} \notin P$ are unreachable. Therefore, the number of reachable states of B is at most $2^{n-1} + 1$.

For the lower bound, we can modify the corresponding construction by Yu et al. [25] for string languages. The proof of Theorem 2.1 of [25] gives an n-state DFA C[3] over alphabet $\Gamma = \{a, b\}$ such that

$$\Gamma^* \cdot L(C) = \{w \in \Gamma^* \mid w = ubv, |v|_a \equiv n - 2 \pmod{n-1}\}.$$

and verifies that the state complexity of $\Gamma^* \cdot L(C)$ is 2^{n-1}. We note that the empty string is not in $\Gamma^* \cdot L(C)$. Thus, when C is interpreted as a tree automaton C' with unary symbols a, b and a nullary symbol σ, a tree automaton recognizing $F_\Sigma \cdot^p_\sigma L(C') \cup \{\sigma\}$ needs one additional state for the leaf symbol σ. □

Now by Lemma 5 (i) and Theorem 3 we get a tight state complexity bound for the corresponding combined operation involving top-down star.

Corollary 1. *If $L \subset F_\Sigma$ is recognized by a DTA with n states, for any $\sigma \in \Sigma_0$, the tree language $(F_\Sigma \cdot^p_\sigma L)^{t,*}_\sigma$ has a DTA with $2^{n-1} + 1$ states and this number of states is necessary in the worst case.*

Theorem 4. *Let A be a DTA with n states and $\sigma \in \Sigma_0$. Then, $(L(A) \cdot^s_\sigma F_\Sigma)^{b,*}_\sigma$ can be recognized by a DTA with $n + 2$ states and this bound can be reached in the worst case.*

Proof. Let $A = (\Sigma, Q_A, Q_{A,F}, g_A)$ be a DTA with n states recognizing the tree language L. We define a DTA $B = (\Sigma, Q_B, Q_{B,F}, g_B)$ for the tree language $(L(A) \cdot^s_\sigma F_\Sigma)^{b,*}_\sigma = L(A) \cdot^s_\sigma F_\Sigma \cup \{\sigma\}$. The following construction assumes that σ_{g_A} is defined and $\sigma_{g_A} \notin Q_{A,F}$. If either of these two conditions is not satisfied, the construction is similar and simpler (in both cases B can do with one fewer state).

Choose

$$Q_B = Q_A \cup \{q_\sigma, q_{\text{dummy}}\}, \quad Q_{B,F} = Q_{A,F} \cup \{q_\sigma\},$$

[3] In the notations of [25], the DFA C is called B.

and the transitions of g_B are defined as below. For $\tau \in \Sigma_0$,

$$\tau_{g_B} = \begin{cases} q_\sigma & \text{if } \tau = \sigma, \\ \tau_{g_A} & \text{if } \tau \neq \sigma \text{ and } \tau_{g_A} \text{ is defined,} \\ q_{\text{dummy}} & \text{otherwise.} \end{cases}$$

Define $g : Q_B \to Q_B$ by setting $g(q_\sigma) = \sigma_{g_A}$ and $g(q) = q$ when $q \neq q_\sigma$. Recall that we assumed that σ_{g_A} is defined. Let q_{final} be an arbitrary but fixed element of $Q_{A,F}$. Now for $\tau \in \Sigma_k, k \geq 1$, and $p_i \in Q_B, i = 1, \ldots, k$, define

$$\tau_{g_B}(p_1, \ldots, p_k) = \begin{cases} q_{\text{final}} & \text{if } \exists j, 1 \leq j \leq k \text{ where } p_j \in Q_{A,F}, \\ \tau_{g_A}(f(p_1), \ldots, f(p_k)) & \text{if } p_1, \ldots p_k \in (Q_A - Q_{A,F}) \cup \{q_\sigma\} \\ & \text{and } \tau_{g_A}(f(p_1), \ldots, f(p_k)) \text{ is defined,} \\ q_{\text{dummy}} & \text{in all other cases.} \end{cases}$$

The DTA B simulates the computation of A up to a point when it reaches a final state, and having reached a final state is marked by entering the state q_{final}. The state q_σ is entered only in a leaf labeled by σ and for transitions on symbols of Σ_k, $k \geq 1$, q_σ is treated as σ_{g_A}. The "copy" of the state σ_{g_A} is needed because B has to accept σ and σ_{g_A} is not accepting. If the computation of A reaches an undefined transition (before entering a final state), B enters the state q_{dummy}. Thus it is clear that B recognizes the set trees having a subtree in $L(A)$ and additionally the tree consisting of the single leaf labeled by σ.

Next we show that the upper bound $n + 2$ is tight. Choose $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$, where $\Sigma_0 = \{c\}$, $\Sigma_1 = \{a\}$ and $\Sigma_2 = \{b\}$. We define a DTA $C = (\Sigma, Q_C, Q_{C,F}, g_C)$, where $Q_C = \{0, 1, \ldots, n-1\}, Q_{C,F} = \{n-1\}$, and the transition function g_C is defined by setting:

$$c_{g_C} = 0, \quad a_{g_C}(i) = i + 1 \pmod{n} \quad \text{for } 0 \leq i \leq n-1.$$

All transitions not listed above are undefined. In particular, note that all transitions for the binary symbol b are undefined. Based on C, we construct a DTA $D = (\Sigma, Q_D, Q_{D,F}, g_D)$ recognizing $(L(A) \cdot_c^s F_\Sigma)_c^{b,*} = L(A) \cdot_c^s F_\Sigma \cup \{c\}$, as described above. Here $Q_D = Q_C \cup \{q_c, q_{\text{dummy}}\}$, $Q_{D,F} = Q_{C,F} \cup \{q_c\}$.

We verify that all states of D are reachable and pairwise inequivalent, and none of the states is a dead state. The state 1 is reached by reading the tree $a(c)$. Then the cyclic transitions on unary symbols a guarantee that states $2, 3, \ldots n$ and 0 are also reachable. The state q_c is reached in a leaf labeled by c and q_{dummy} is reachable because C has undefined transitions.

States $0 \leq i < j \leq n-1$ are not equivalent because by reading $n-1-i$ unary symbols a the state i ends in the accepting state $n-1$ and by reading the same sequence of unary symbols j does not enter an accepting state. By the same reasoning q_c is not equivalent to any state $1 \leq j \leq n$. The state q_c is not equivalent with 0 because the former is a final state and the latter is not. The state q_{dummy} cannot reach a final by reading a sequence of a's while all other states have this

property. Finally to verify that none of the states is a dead state, above we have already observed that the states $0 \leq i \leq n-1$ and q_c can reach a final state by reading a sequence of a's. According to the definition of the transitions of D, $b_{g_D}(q_{\text{dummy}}, n-1) = n-1$ and it follows that also q_{dummy} is not a dead state. (Note that in the DTA D we must have $q_{\text{final}} = n-1$ since $n-1$ is the only final state of C.)

We have verified that the minimal DTA for $L(A) \cdot_c^s F_\Sigma \cup \{c\}$ has $n+2$ states and this concludes the proof. □

In the construction used for the lower bound of Theorem 4, the symbol b of rank two has no defined transitions in the original DTA C. However, it can be noted that the tight bound cannot be reached by tree languages over a ranked alphabet that has no symbols of rank greater than one. If the ranked alphabet has only unary and nullary symbols, in the DTA B constructed to recognize the tree language $(L(A) \cdot_\sigma^s F_\Sigma)_\sigma^{b,*}$ the state q_{dummy} will always be a dead state.

Again using Lemma 5 (ii) and Theorem 4 we get a tight bound for the same combined operation involving top-down star:

Corollary 2. *For a tree language L recognized by a DTA with n states and $\sigma \in \Sigma_0$, the tree language $(L \cdot_\sigma^s F_\Sigma)_\sigma^{t,*}$ has a DTA with $n+2$ states and $n+2$ states is needed in the worst case.*

For establishing an upper bound for the combined operations $(L \cdot_\sigma^p F_\Sigma)_\sigma^{b,*}$ and $(L \cdot_\sigma^p F_\Sigma)_\sigma^{t,*}$ we first consider a construction for the parallel concatenation of L and F_Σ. If A is an n-state DFA on strings over alphabet Γ, the language $L(A) \cdot \Gamma^*$ can be recognized by a DFA with n states. For the parallel concatenation of an n-state tree language and F_Σ we use $2n$ states.

Lemma 6. *Let A be a DTA with n states and f final states and $\sigma \in \Sigma_0$. Then, $L(A) \cdot_\sigma^p F_\Sigma$ can be recognized by a DTA with $2n+1-f$ states.*

Proof. Let $A = (\Sigma, Q_A, Q_{A,F}, g_A)$. We construct a DTA $B = (\Sigma, Q_B, Q_{B,F}, g_B)$ for the tree language $L(A) \cdot_\sigma^p F_\Sigma$. Note that if $\sigma \in L(A)$, then $L(A) \cdot_\sigma^p F_\Sigma = F_\Sigma$. Without loss of generality we can assume that $\sigma_{g_A} \notin Q_{A,F}$. Choose

$$Q_B = \{0,1\} \times (Q_A - Q_{A,F}) \ \cup \ \{0\} \times (Q_{A,F} \cup \{q_{\text{Adead}}\}),$$

where q_{Adead} is a new element not in Q_A, $Q_{B,F} = \{(0,q) \mid q \in Q_A \cup \{q_{\text{Adead}}\}\}$ and the transitions of g_B are defined as below. We set $\sigma_{g_B} = (1, \sigma_{g_A})$ if σ_{g_A} is defined, and σ_{g_B} is undefined otherwise. For $\tau \in \Sigma_0$, $\tau \neq \sigma$,

$$\tau_{g_B} = \begin{cases} (0, \tau_{g_A}) & \text{if } \tau_{g_A} \text{ is defined,} \\ (0, q_{\text{Adead}}) & \text{if } \tau_{g_A} \text{ is not defined.} \end{cases}$$

For $\tau \in \Sigma_k, k \geq 1$, and $x_1, \ldots, x_k \in \{0,1\}$, $q_1, \ldots, q_k \in Q_A \cup \{q_{\text{Adead}}\}$ we define $\tau_{g_B}((x_1,q_1), \ldots, (x_k,q_k))$ to be

(i) $(1, \tau_{g_A}(q_1, \ldots, q_k))$ if there exists $1 \leq i \leq k$ such that $x_i = 1$ and $\tau_{g_A}(q_1, \ldots, q_k) \in Q - Q_{A,F}$,

(ii) $(0, \tau_{g_A}(q_1, \ldots, q_k))$ if $\tau_{g_A}(q_1, \ldots, q_k) \in Q_{A,F}$,

(iii) $(0, \tau_{g_A}(q_1, \ldots, q_k))$ if $x_i = 0$, $i = 1, \ldots, k$ and $\tau_{g_A}(q_1, \ldots, q_k))$ is defined,

(iv) $(0, q_{\mathrm{Adead}})$ if $\tau_{g_A}(q_1, \ldots, q_k)$ is undefined and $x_1 = \ldots = x_k = 0$,

(v) undefined in all the remaining cases.

Note that in the above definitions if some q_i is q_{Adead}, the transition $\tau_{g_A}(q_1, \ldots, q_k)$ is naturally undefined.

We note that the tree language $L(A) \cdot^p_\sigma F_\Sigma$ consists of all Σ-trees t that have the property that any leaf labeled by σ must belong to a subtree of t that is in $L(A)$. The DTA B checks this property as follows. The second components of the states simulate the computation of A and the bit in the first component keeps track of whether or not the current subtree has a leaf labeled by σ that "was not part of a subtree" belonging to $L(A)$. More precisely, suppose that the computation reaches the root of t in state (x, q). If $x = 1$, this indicates t had a leaf ℓ labeled by σ and the computation from ℓ to the root of t has not passed through a final state of A. Note that in the transitions of B when the second component becomes an element of $Q_{A,F}$ the first component is always reset to 0, that is, pairs of the form $(1, q)$, $q \in Q_{A,F}$, are not used as states of B. If the second component of the state is q_{Adead}, this indicates that the computation of A on the current subtree t is undefined. In this situation if t contains a leaf labeled by σ, t cannot be a subtree of $L(A) \cdot^p_\sigma F_\Sigma$ and the state $(1, q_{\mathrm{Adead}})$ is not in Q_B.

The claim follows since $|Q_B| = 2 \cdot |Q_A| + 1 - |Q_{A,F}|$. □

Now combining Lemma 6 with, respectively, Theorem 1 and Theorem 2 we get the following upper bounds for the state complexity of bottom-up or top-down star of a tree language $L \cdot^p_\sigma F_\Sigma$. Note that the bound of Lemma 6 reaches the worst case $2n$ when the DTA has exactly one final state.

Proposition 1. *Let A be a DTA with n states and $\sigma \in \Sigma_0$. Then $(L(A) \cdot^p_\sigma F_\Sigma)^{b,*}_\sigma$ can be recognized by a DTA with $(4n + 3) \cdot 4^{n-1}$ states.*

The tree language $(L(A) \cdot^p_\sigma F_\Sigma)^{t,*}_\sigma$ can be recognized by a DTA with $3 \cdot 4^{n-1}$ states.

We do not know whether the bounds of Proposition 1 are optimal. Finally, the bottom-up or top-down star of the sequential concatenation of F_Σ with a regular tree language L, $F_\Sigma \cdot^s_\sigma L$, seem to be the most problematic of the combined operations involving Kleene star and concatenation with F_Σ. For these operations we know only trivial upper bounds implied by the state complexity of sequential concatenation and the corresponding star operation.

5 Conclusion

In the last section we have considered the state complexity of star–of–concatenation in the special case where one of the argument tree languages consists of the set of all trees. The precise state complexity of star–of–concatenation remains open for

general tree languages. For references dealing with the string case the reader may consult [2].

For top-down and bottom-up star we have established the precise worst case state complexity. The lower bound construction for Kleene-star in [24] uses a two-letter alphabet, and hence the worst-case state complexity of top-down star can be achieved over a ranked alphabet with two unary and one nullary symbol. It is clear that one unary and one nullary symbol is not sufficient because the state complexity of Kleene-star for string languages over a one-letter alphabet is $(n-1)^2+1$ [24]. With one binary symbol ω and one nullary symbol σ, we can encode strings over a two letter alphabet as trees "built up" from elements $\omega(\sigma, x)$ and $\omega(x, \sigma)$. In this way one clearly gets an exponential lower bound construction, however, we do not know whether one binary and one nullary symbol is sufficient to reach the precise bound of Theorem 2.

Our lower bound construction for Theorem 1 uses a ranked alphabet of six symbols. The state complexity for bottom-up star is of a different order of magnitude than the corresponding bound for string languages. This means that the worst-case constructions essentially need to rely on "tree properties" and finding the minimal alphabet size remains an open question.

References

[1] Comon, H., Dauchet, M., Gilleron, R., Jacquemard, F., Lugiez, D., Löding, C., Tison, S., Tommasi, M.: *Tree Automata Techniques and Applications,* electronic book available at `tata.gforge.inria.fr`, 2007.

[2] Cui, B., Gao, Y., Kari, L., Yu, S.: State complexity of combined operations with two basic operations. Theoret. Comput. Sci. 437 (2012) 82–102.

[3] Eom, H.-S., Han, Y.-S., Ko, S.-K.: State complexity of subtree free regular tree languages, *Proc. DCFS'15*, LNCS 8031, Springer, 2013, pp. 66–77.

[4] Gao, Y., Moreira, N., Reis, R., Yu, S.: A review on state complexity of individual operations. To appear in *Computer Science Review.* (Available at `www.dcc.fc.up.pt/dcc/Pubs/TReports/TR11/dcc-2011-08.pdf`)

[5] Gécseg, F., Steinby, M.: *Tree automata,* Akadémiai Kiadó, 1984.

[6] Gécseg, F., Steinby, M.: Tree languages, *Handbook of Formal Languages,* Vol. III, (G. Rozenberg, A. Salomaa Eds.), Springer, 1997, pp. 1–68.

[7] Han, Y.-S., Salomaa, K.: Nondeterministic state complexity of nested word automata. Theoret. Comput. Sci. 410 (2009) 2961–2971.

[8] Holzer, M., Kutrib, M.: Descriptional and computational complexity of finite automata — A survey. Inf. Comput. 209 (2011) 456–470.

[9] Holzer, M., Salomaa, K., Yu, S.: On the state complexity of k-entry deterministic finite automata. J. Autom., Lang. and Combinatorics 6 (2001) 453–466.

[10] Jirásková, G., Okhotin, A.: On the state complexity of star of union and star of intersection. Fund. Informaticae 109 (2011) 161–178.

[11] Ko, S.-K., Lee, H.-R., Han, Y.-S.: State complexity of regular tree languages for tree pattern matching, *Proc. of DCFS'14*, LNCS 8614, Springer, 2014, pp. 246–257.

[12] Kutrib, M., Pighizzini, G.: Recent trends in descriptional complexity of formal languages. *Bulletin of the EATCS* 111 (2013) 70–86.

[13] Lupanov, O.B.: A comparison of two types of finite sources. *Problemy Kibernetiki* 9 (1963) 328–335.

[14] Martens, W., Niehren, J.: On the minimization of XML schemas and tree automata for unranked trees. J. Comput. System Sci. 73 (2007) 550–583.

[15] Maslov, A.N.: Estimates of the number of states of finite automata, *Dokl. Akad. Nauk. SSSR,* **194** (1970) Soviet Math. Dokl. 11 (1970) 1373–1375.

[16] Meyer, A.R., Fischer, M.J.: Economy of description by automata, grammars and formal systems. Proc. SWAT (FOCS), IEEE Computer Society (1971) 188–191.

[17] Okhotin, A., Salomaa, K.: Descriptional complexity of unambiguous input-driven pushdown automata. Theoret. Comput. Sci. 566 (2015) 1–11.

[18] Piao, X., Salomaa, K.: State complexity of the concatenation of regular tree languages. Theoret. Comput. Sci. 429 (2012) 273–281

[19] Piao, X., Salomaa, K.: Transformations between different models of unranked bottom-up tree automata. Fund. Informaticae 109 (2011) 405–424.

[20] Piao, X., Salomaa, K.: State complexity of star and quotient operation for unranked tree automata, School of Computing, Queen's University Technical Report No. 2011-577 (19 pp.), 2011

[21] Salomaa, A., Salomaa, K., Yu, S.: State complexity of combined operations. Theoret. Comput. Sci. 383 (2007) 140–152.

[22] Schwentick, T.: Automata for XML, — a survey. J. Comput. System Sci. 73 (2007) 289–315.

[23] Shallit, J.: *A Second Course in Formal Languages and Automata Theory,* Cambridge University Press, 2009.

[24] Yu, S.: Regular languages, in: *Handbook of Formal Languages,* Vol. I, (G. Rozenberg, A. Salomaa, Eds.), Springer, 1997, pp. 41–110.

[25] Yu, S., Zhuang, Q., Salomaa, K.: The state complexity of some basic operations on regular languages. Theoret. Comput. Sci. 125 (1994) 315–328.

Received 2nd June 2015

Acta Cybernetica **22** (2015) 423–433.

On a Parity Based Group Testing Algorithm*

Sándor Z. Kiss†, Éva Hosszu‡, Lajos Rónyai§, and János Tapolcai‡

This paper is dedicated to the memory of Professor Ferenc Gécseg

Abstract

In traditional Combinatorial Group Testing the problem is to identify up to d defective items from a set of n items on the basis of group tests. In this paper we describe a variant of the group testing problem above, which we call *parity group testing*. The problem is to identify up to d defective items from a set of n items as in the classical group test problem. The main difference is that we check the parity of the defective items in a subset. The test can be applied to an arbitrary subset of the n items with two possible outcomes. The test is positive if the number of defective items in the subset is odd, otherwise it is negative. In this paper we extend Hirschberg et al.'s method to the parity group testing scenario.

Keywords: combinatorial group testing

1 Introduction

1.1 Motivation

Dealing with errors during transmission has been a long-standing problem of communication theory. Numerous error scenarios have been considered, mostly focusing on cases when the channel is unreliable. In [8] Hachem et al. proposed a novel possibility: what if the encoder itself is introducing uncertainty?

There are several causes as to why an encoder might behave in a faulty manner [8]. First, the physical device implementing the encoder might be faulty, causing the encoder to have faults itself. Second, due to ever-reducing chip size, soft errors

*The work of János Tapolcai was partially supported by the Hungarian Scientific Research Fund (grant No. OTKA 108947). The work of Lajos Rónyai was supported by the Hungarian Scientific Research Fund (grant No. OTKA NK 105645).

†Budapest University of Technology and Economics, Department of Algebra E-mail: kisspest@cs.elte.hu

‡MTA-BME Future Internet Research Group, Budapest University of Technology and Economics (BME) E-mail: {hosszu,tapolcai}@tmit.bme.hu

§Computer and Automation Research Institute Hungarian Academy of Sciences and Budapest University of Technology and Economics, Department of Algebra, E-mail: ronyai@sztaki.hu

$$\begin{array}{c} \\ p_1 \\ p_2 \\ p_3 \end{array} \begin{array}{c} \begin{array}{ccccccc} c_1 & c_2 & c_3 & c_4 & c_5 & c_6 & c_7 \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix} \end{array}$$

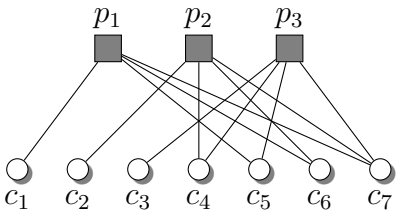

Figure 1: The generator matrix and the Tanner graph of the $(7, 3)$ dual Hamming code.

in processing and storage are becoming more and more frequent [13]. Third, with the scaling of technology, device degradation and variability in transistor design may also cause unreliable behaviour [3]. Lastly, errors might happen during distributed encoding when physically separated devices are connected through a noisy channel, as in sensor networks [2].

In this work we adapt their fault model. Let us consider the Tanner-type factor graph defined as below. For a given $k \times n$ linear code G and $(n-k) \times n$ parity check matrix H the *Tanner graph* is the following. The Tanner graph $T = (\{V_1, V_2\}, E)$ consists of node set $V_1 \dot\cup V_2$, where $|V_1| = k$ and $|V_2| = n$; and for $i \in \{1, \ldots, k\}, j \in \{1, \ldots, n\}$ edge set $E = \{\{v_1^i, v_2^j\} : v_1^i \in V_1, v_2^j \in V_2, G[i, j] = 1\}$.

Note that as the Tanner graph is defined by the generator matrix, it is not necessarily unique to the code.

Let us model the faults as edges getting erased in the factor graph of G, which reveal themselves as bits getting flipped $1 \to 0$. It is assumed that due to the edge erasures every bit that is 1 may get flipped to 0 independently from each other with probability p.

Assuming that the original generator matrix is known to both the receiver and the transmitter, an easy way to check against erasures would be to send the unit vectors of length k as test messages. In this case when sending the i^{th} unit vector the receiver would receive the i^{th} row of the generator thus enabling to detect any number of faults after getting all the messages – as many as the number of rows in the generator. The natural question follows: can one do better?

In this work we investigate what one can do to check whether the encoder itself is introducing uncertainty. Hachem et al. [8] considered the problem of introducing enough redundancy so as to counteract the effects of a faulty encoder. The problem we address in this paper is how one would go about discovering the locations of these erasures.

1.2 Introducing Parity Group Testing

The traditional problem in group testing is the following. Let S be a set of items with n elements, some of them (say, at most d) are possibly defective. For simpler notation we assume that $S = \{1, 2, \ldots, n\}$. We intend to find the defective items via *group tests*. A group test is a subset T of S; testing T has two possible outcomes.

It is positive if there is at least one defective item in T and negative otherwise. The tests may be executed either in an adaptive manner, taking the preceding tests' outcome into account when designing the next one, or non-adaptively, when all tests are to be determined at the start. In this paper we consider the non-adaptive version of the problem.

The main objective of any combinatorial group testing (CGT) scheme is to find the defective elements via such group tests efficiently. Efficiency may be measured in different ways, a prevalent goal is to try and minimize the number of subsets T to be tested. There is rich literature on the subject, for further details we refer the reader to [4, 10, 11].

Translating this concept to binary linear encoders goes as follows. The set of items are all the bits that could get erased, the 1s in the generator matrix. A test would be a message, which gets evaluated based on whether it differs from what we were supposed to receive or not – assuming that the generator matrix of the code is known to both the receiver and the transmitter. The items included in a test are the ones from every row where there is a 1 in the test message, so individual testing of the items would be to send messages that contain only a single 1 in them, i.e. the unit vectors. Testing a pool of potential erasures is to send a message that contains more than just one bit that is 1.

Let us present an illustrative example. Let G be the generator matrix for the (7,4)–Hamming code, known to both the transmitter and the receiver. Suppose the erasures denoted by bold **0**'s on Figure 2 happen. Sending the unit vectors of length 4 would display the current state of G row-by-row on the receiver side, making it possible to diagnose any number of faults using 4 messages.

However, the erasures cancel each other out if we send a message containing more than just one bit that is 1 and they hit an even number of erasures. For example let us send the message $(1, 1, 0, 0)$ using G' depicted on Figure 2b. The received word would be $(0, 0, 0, 1, 0, 1, 1)$ whereas the correct word we should receive with an erasure-free received word is $(1, 0, 0, 0, 0, 1, 1)$. This reveals that there are erasures in the first and fourth column but the two erasures in the second column don't show up.

Motivated by this observation, we define *parity group testing* as follows. In the parity group testing problem the aim again is to find at most d defectives in an n-element set S. However, the two outcomes of a test $T \subseteq S$ are changed: instead of revealing the presence of defectives in T the result of a test will now show whether there is an *odd* or *even* number of defective items in T, hence the name parity testing. Our aim is for given set size n and maximum number of defectives d identify all the defective items such that the number of necessary parity group tests is small.

$$\begin{pmatrix}1\\1\\0\\0\end{pmatrix} \quad \begin{pmatrix}1&1&0&1&0&0&1\\0&1&0&1&0&1&0\\1&0&0&1&1&0&0\\1&1&1&0&0&0&0\\1&0&0&0&0&1&1\end{pmatrix}$$

(a) The generator matrix G sending the message $(1, 1, 0, 0)$.

$$\begin{pmatrix}1\\1\\0\\0\end{pmatrix} \quad \begin{pmatrix}\mathbf{0}&\mathbf{0}&0&\mathbf{0}&0&0&1\\0&\mathbf{0}&0&1&0&1&0\\\mathbf{0}&0&0&1&1&0&0\\1&1&1&0&0&0&0\\0&0&0&1&0&1&1\end{pmatrix}$$

(b) The erasure-stricken matrix G' sending the message $(1, 1, 0, 0)$.

$$\begin{array}{rc} & \begin{pmatrix}1&0&0&0&0&1&1\end{pmatrix}\\ \text{XOR} & \begin{pmatrix}0&0&0&1&0&1&1\end{pmatrix}\\ \hline & \begin{pmatrix}1&0&0&1&0&0&0\end{pmatrix}\end{array}$$

(c) The resulting group test reveals an error in the first column by taking the XOR of the received words.

Figure 2: An example of group tests translated to linear encoders.

2 A Chinese Remainder Theorem based CGT Algorithm

In this section first we recap a previous CGT algorithm our parity group testing constructions are based on, then we describe our algorithms for identifying faulty items in the parity setting. We assume the underlying set S to be $\{1, \ldots, n\}$ and that there are at most d faulty items (unless stated otherwise).

Eppstein, Goodrich and Hirschberg [5] provided a non-adaptive combinatorial group testing algorithm based on the Chinese Remainder Theorem. First a sequence of pairwise coprime positive integers $\{p_1, p_2, \ldots, p_k\}$ is selected such that

$$n^d \leq P = \prod_{i=1}^{k} p_i.$$

In this setting the the total number of tests would be

$$t(n, d) = \sum_{i=1}^{k} p_i.$$

We may assume that $p_1 < p_2 < \cdots < p_k$. The first group test X contains the numbers a where $a \equiv 0 \pmod{p_1}$ holds, while the second contains the numbers b satisfying $b \equiv 1 \pmod{p_1}$, and so on, till all remainders for each p_i are taken for $i = 1, \ldots, k$.

2.1 Constructive Algorithm to Find the Solution for Single Defective items

Note that if there is at most one defective item, then parity group testing is the same as the classical group testing problem, i.e., if the set X contains odd number of defective items, then it follows that the only defective item is in X, otherwise X does not contain the defective item.

Let a_i denote the remainder of a single item $x \in S$ for p_i. The task is to find the number x which satisfies the following system of congruences:

$$x \equiv a_i \pmod{p_i} \tag{1}$$

for $i = 0, \ldots, k$.

For each i the integers p_i and $\prod_{j \neq i} p_j$ are relatively prime. Using the extended Euclidean algorithm we can find integers r_i and q_i such that

$$r_i p_i + q_i \prod_{j \neq i} p_j = 1.$$

Then, choosing $e_i = q_i \prod_{j \neq i} p_j$, x can be reconstructed by

$$x = \sum_{i=1}^{k} a_i e_i \pmod{\prod_j p_j} \tag{2}$$

which satisfies (1). This well known scheme of reconstruction from Chinese Remainders can be summarized as follows.

Algorithm 1 Chinese Remainder

Input: $(p_1, \ldots, p_k)$, $(a_1, \ldots, a_k)$

for $i = 1$ to k **do**

Compute

$$N_i = \prod_{j \neq i} p_j,$$

$$q_i = N_i^{-1} \pmod{p_i}.$$

end for

Compute

$$x = \sum_{i=1}^{k} a_i q_i N_i \pmod{p_1 p_2 \cdots p_k}.$$

2.2 Constructive Algorithm to Find the Solution for d defective items in the parity setting

Let $x_1, \ldots, x_d$ denote the defective items, where $d > 1$.

The following simple fact shows that the defective items can be well separated in the different residue classes.

Claim 1. *Let $e_1, \ldots, e_v$ be pairwise coprime positive integers. If $v \geq \binom{d}{2} \log_2 n$, then there exists an e_i, where $1 \leq i \leq v$ such that $x_1, \ldots, x_d$ lie in different residue classes modulo e_i.*

Proof. We prove the statement by contradiction. Assume that $1 \leq x_1 < \ldots < x_d \leq n$, and for any $1 \leq i \leq v$ there are at least two elements among $x_1, \ldots, x_d$ such that they are in the same residue classes modulo e_i. In other words for all $1 \leq i \leq v$, there exist $1 \leq l < m \leq d$ such that $e_i | x_m - x_l$. There may be at most $\binom{d}{2}$ pairs of the last type, hence by the pigeonhole principle there exist $1 \leq r < s \leq d$ such that for at least $c \geq \log_2 n$ different indices j we have $e_j | x_s - x_r$. As e_i's are pairwise coprime, it follows that $\prod e_j | (x_s - x_r)$, but $n \leq 2^c \leq \prod e_j | (x_s - x_r) < n$ which is a contradiction. (Here the product is over the indices j such that $e_j | x_s - x_r$.) □

If we set $k \geq \binom{d}{2} \log_2 n + d \log_2 n + 1$, it follows from the above Claim that there exists pairwise coprime numbers $p_1, \ldots, p_t$ among the numbers $p_1, \ldots, p_k$ such that $p_1 \cdots p_t \geq n^d$ and $x_1, \ldots, x_d$ lie in different residue classes modulo p_i, where $1 \leq i \leq t$. This means that parity testing with the integers $p_1, \ldots, p_t$ the positive outcome (i.e., when the parity of the defective items is odd in a residue class modulo p_i) implies that there is exactly one defective item in the corresponding residue class. Please note that such a collection $p_1, \ldots, p_t$ can be efficiently selected from $p_1, \ldots, p_k$.

Let $y_i^{(1)}, \ldots, y_i^{(d)}$ denote the remainders of the d defective items $x_1, \ldots, x_d \in S$ modulo p_i. Recall that we selected the moduli p_i in such a way that

$$n^d \leq P = \prod_{i=1}^{t} p_i.$$

The task is to find the numbers $x_1, \ldots, x_d$ which satisfy the following system of congruences:

$$x_1 \equiv y_i^{(1)} \pmod{p_i}, \quad \ldots, \quad x_d \equiv y_i^{(d)} \pmod{p_i}$$

for all $1 \leq i \leq t$.

Please note that for an i the residues $y_i^{(j)}$ are pairwise different for $j = 1, \ldots, d$. Having the numbers $y_i^{(j)}$ at hand, we can calculate the residues of the elementary symmetric polynomials[1] of $x_1, \ldots, x_d$ modulo all the p_i by using Algorithm 3:

$$\sigma_1(x_1, \ldots, x_d) \equiv a_1^{(1)} \pmod{p_1}, \ldots, \sigma_1(x_1, \ldots, x_d) \equiv a_t^{(1)} \pmod{p_t};$$

$$\vdots$$

$$\sigma_d(x_1, \ldots, x_d) \equiv a_1^{(d)} \pmod{p_1}, \ldots, \sigma_d(x_1, \ldots, x_d) \equiv a_t^{(d)} \pmod{p_t};$$

By using the Chinese remainder theorem we can calculate

$$\sigma_1(x_1, \ldots, x_d) \equiv A_1 \pmod{P}, \ldots, \sigma_d(x_1, \ldots, x_d) \equiv A_d \pmod{P}.$$

[1] For details, see the Appendix.

As $P \geq n^d$ and

$$0 < \sigma_1(x_1, \ldots, x_d), \ldots, \sigma_d(x_1, \ldots, x_d) < n^d$$

the following equalities hold.

$$\sigma_1(x_1, \ldots, x_d) = A_1, \ldots, \sigma_d(x_1, \ldots, x_d) = A_d \ .$$

It is easy to see that the roots of the polynomial

$$f(w) = w^d - \sigma_1 w^{d-1} + \sigma_2 w^{d-2} - .... + (-1)^d \sigma_d$$

are $x_1, \ldots, x_d$. We can find the roots of f by using the root finder method [9]. The essence of this method is to isolate the roots by using the Sturm theorem and we can find the roots applying the bisection method (binary search). More formally we have the following algorithm.

Algorithm 2 Parity based Chinese Remainder Sieve algorithm

Input: $y_i^{(1)}, \ldots, y_i^{(d)}$ for all $1 \leq i \leq t$, $p_1, \ldots, p_t$

1: **for** $j = 1$ to d **do**
2: **for** $i = 1$ to t **do**
3: $\sigma_j(y_i^{(1)}, \ldots, y_i^{(d)}) = a_i^{(j)} \pmod{p_i}$
4: **end for**
5: $A_j = \text{ChineseRemainder}(a_1^{(j)}, \ldots, a_t^{(j)}, p_1, \ldots, p_t)$
6: **end for**
7: Set $f(z) = z^d + \sum_{l=1}^{d} (-1)^l A_l z^{d-l}$
8: Compute $(x_1, x_2, \ldots, x_d) = \text{Root Finder } (f(z))$

3 Analysis

In this section we will give a brief analysis of the running time of our algorithm and an upper bound for the number of test required to identify the defective items as well. Throughout the remaining part of this section $\log n$ denotes the natural logarithm i.e., the logarithm to the base e.

3.1 Number of tests

Let $t(n, d)$ denote the number of tests constructed in the Chinese Remainder Sieve discovered by Hirschberg et al. They proved that the d defective items could be identified using the number of tests

$$t(n, d) < \frac{\lceil 2d \log n \rceil^2}{2 \log \lceil 2d \log n \rceil} \Big(1 + \frac{1.2762}{\log \lceil 2d \log n \rceil} \Big).$$

As noted in the introduction, in our case the number of required tests is

$$t(n, d) = \sum_{i=1}^{k} p_i.$$

To simplify the calculations we can assume that the p_i's are primes. Let q_i denote the ith largest prime. It follows that we have to estimate

$$\sum_{i=1}^{k} q_i.$$

It is well known [7] that $q_k = O(k \log k)$ which implies that

$$\sum_{i=1}^{k} q_i = O(k^2 \log k).$$

In our case we can choose $k = \binom{d}{2} \log_2 n + d \log_2 n + 1 = \frac{d(d+1)}{2} \log_2 n + 1$, thus we have the following upper bound to the number of tests in the parity case:

$$t(n,d) = O\Big(d^4 \log^2 n \cdot \log d + d^4 \log^2 n \cdot \log\log n\Big).$$

3.2 Running time

Claim 2. *The Parity based Chinese Remainder Sieve algorithm finds the defective items by using $O(d^{10} \log^3 n)$ bit operations. This is in addition to the cost of the tests.*

Proof. The Parity based Chinese Remainder Sieve algorithm contains four steps. In the first step it determined the residues y_i^j. They are essentially the outcomes of the tests. In the second step, it computes the elementary symmetric polynomials, in the third step it uses the Chinese remainder theorem, and finally it determines the roots of the corresponding polynomial.

In Algorithm 3 we compute the symmetric polynomials recursively. In the rth step there are $r-1$ additions and $r-1$ multiplications, thus we can compute all symmetric polynomials by using $1 + \ldots + (d-1)$ additions and multiplications. As $1 \le x_1, \ldots, x_d \le n$, one addition needs $O(\log n)$ bitoperations, and one multiplication requires $O(\log^2 n)$ bit operations, thus the total cost of Algorithm 3. is $O(d^2 \log^2 n)$ bit operations.

In this paragraph we analyze the Chinese remaindering process (Algorithm 1.) It is well known [1] that Chinese remaindering requires $O(\log^2 P)$ bitoperations. It is easy to see [16] that

$$\log P \le \sum_{i=1}^{k} \log q_i \le \pi(q_k) \log q_k = k \log q_k,$$

where $\pi(x)$ denotes the number of primes up to x. It is well known [7] that the kth prime number is $O(k \log k)$, thus we have

$$\log P = O(k(\log k + \log\log k)) = O(k \log k).$$

We know that

$$k = O(d^2 \log n),$$

which implies $k \log k = O(d^2 \log n(\log d + \log\log n))$. It follows that the total cost is $O(d^4 \log^2 n \cdot (\log^2 d + (\log\log n)^2))$. Since the number of systems of congruences is d, computing the A_j's in the Chinese Remainder Filter needs $O(d^5 \log^2 n(\log^2 d + (\log\log n)^2))$ bit operations.

In the last step we have to determine the roots of the polynomial $f(z)$. For a polynomial $f(z) = a_d z^d + \ldots + a_1 z + a_0$ let

$$K = \sum_{i=0}^{d} |a_i|.$$

It is clear that all coefficients of our polynomial are at most n^d, which implies that $K < dn^d$. It follows from [6] that the running time of Heindel's algorithm is $O(d^{10} + d^7 \log^3 K)$. We have to use the bisection method at most $d-1$ times, which requires $O(d \log n)$ operations, because the length of each interval is at most n. Thus the total cost to determine all roots requires at most $O(d^{10} + d^{10} \log^3 n + d \log n) = O(d^{10} \log^3 n)$ bitoperations. This implies that the total cost of the Chinese Remainder Filter Algorithm is $O(d^2 \log^2 n + d^5 \log^2 n(\log^2 d + (\log\log n)^2) + d^{10} \log^3 n) = O(d^{10} \log^3 n)$ bit operations. □

Please note that there is a more sophisticated algorithm than Heindel's method, it can be found in [15]. The running time of this algorithm is better than Heindel's algorithm.

4 Conclusions

Motivated by the problem of error location in a linear encoder in this paper we introduced a novel variant of a classic combinatorial search task called *parity group testing*. After presenting the basic framework we showed how to adapt the Chinese Remainder Theorem based search algorithm to our scenario such that d defectives can be found in a set of n elements using $O\Big(d^4 \log^2 n \cdot \log d + d^4 \log^2 n \cdot \log\log n\Big)$ parity group tests, using $O(d^{10} \log^3 n)$ bit operations.

References

[1] P. C. van Oorschot A. J. Menezes and S. A. Vanstone. *Handbook of Applied Cryptography*, volume 4. CRC Press, 1996.

[2] Ian F Akyildiz, Weilian Su, Yogesh Sankarasubramaniam, and Erdal Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.

[3] Aris Christou. *Electromigration and Electronic Device Degradation.* Wiley-Interscience, 1994.

[4] Ding Zhu Du and Frank Hwang. *Combinatorial group testing and its applications*. World Scientific, 1993.

[5] David Eppstein, Michael T Goodrich, and Daniel S Hirschberg. Improved combinatorial group testing algorithms for real-world problem sizes. *SIAM Journal on Computing*, 36(5):1360–1375, 2007.

[6] R. Loos G. E. Collins. Polynomial real root isolation by differentation. In *Proceedings of the 1976 ACM Symposium on Symbolic and Algebraic Computation*, pages 15–20. ACM, 1976.

[7] E. Kowalski H. Iwaniec. *Analytic Number Theory*, volume 53. American Mathematical Society, 2004.

[8] Jad Hachem, I-Hsiang Wang, Christina Fragouli, and Suhas Diggavi. Coding with encoding uncertainty. In *IEEE International Symposium on Information Theory Proceedings (ISIT)*, pages 276–280. IEEE, 2013.

[9] Lee E. Heindel. Integer arithmetic algorithms for polynomial real zero determination. *J. ACM*, 18(4):533–548, October 1971.

[10] FK Hwang. A method for detecting all defective members in a population by group testing. *Journal of the American Statistical Association*, 67(339):605–608, 1972.

[11] FK Hwang and VT Sós. Non-adaptive hypergeometric group testing. *Studia Sci. Math. Hungar*, 22:257–263, 1987.

[12] Hao Jiang, Stef Graillat, and Roberto Barrio. Accurate and fast evaluation of elementary symmetric functions. In *IEEE Symposium on Computer Arithmetic*, pages 183–190, 2013.

[13] Michael Nicolaidis. *Circuit-Level Soft-Error Mitigation*. Springer, 2011.

[14] Viktor V. Prasolov. *Polynomials*. Springer, 2004.

[15] Michael Sagraloff and Kurt Mehlhorn. Computing real roots of real polynomials. *Journal of Symbolic Computation*, 2015.

[16] G. Tenenbaum. *Introduction to analytic and probabilistic number theory*, volume 46. Cambridge University Press, 1995.

Appendix

We need the following facts about polynomials [14]. For $m \geq 0$, let

$$\sigma_m = \sigma_m(t_1, \ldots, t_d) = \sum_{1 \leq j_1 < j_2 < \ldots < j_m \leq d} t_{j_1} \cdot \ldots \cdot t_{j_m}$$

be the m^{th} elementary symmetric polynomial of $t_1, \ldots, t_d$.

We can compute the elementary symmetric polynomials by using the following algorithm [12].

Algorithm 3 Elementary Symmetric Polynomial Calculator

Input: $X = (x_1, \ldots, x_d)$ and m
Output: all the elementary symmetric polynomials $\sigma_1, \ldots, \sigma_d$

1: function $\sigma_m^{(d)} = \texttt{SumESF}(X, m)$
2: $\sigma_0^{(i)} = 1, 1 \leq i \leq d-1; \sigma_j^{(i)} = 0, j > i; \sigma_1^{(1)} = x_1$
3: **for** $i = 2$ to d **do**
4: **for** $j = 1$ to i **do**
5: $\sigma_j^{(i)} = \sigma_j^{(i-1)} + x_i \sigma_{j-1}^{(i-1)}$
6: **end for**
7: **end for**

It is also well known [14] that if we have a polynomial $p(x)$, where α_i denotes its coefficients and β_is are the roots of $p(x)$,

$$p(x) = x^d + \ldots + \alpha_{d-1}x + \alpha_d = (x - \beta_1) \ldots (x - \beta_d),$$

then we have $\alpha_i = (-1)^{d-i}\sigma_i(\beta_1, \ldots, \beta_d)$.

Received 15th June 2015

Acta Cybernetica **22** (2015) 435–483.

Weighted First-Order Logics over Semirings*

Eleni Mandrali† and George Rahonis‡

Dedicated to the memory of Ferenc Gécseg

Abstract

We consider a first-order logic, a linear temporal logic, star-free expressions and counter-free Büchi automata, with weights, over idempotent, zero-divisor free and totally commutative complete semirings. We show the expressive equivalence (of fragments) of these concepts, generalizing in the quantitative setup, the corresponding folklore result of formal language theory.

1 Introduction

The expressive equivalence of monadic second-order logic and finite automata over finite words was established in [5, 16] and over infinite words in [6]. Droste and Gastin, in [8] (cf. also [9]), introduced a weighted monadic second-order logic over semirings and showed that sentences from a fragment of this logic, interpreted over finite words, are equivalent to weighted automata. A corresponding result for infinite words was stated in [13]. Recently in [12], the authors extended the expressive equivalence of monadic second-order logic and automata over more general structures, namely valuation monoids. On the other hand, first-order (*FO* for short) logic (i.e., the logic obtained from monadic second-order one by relaxing second-order quantifiers) is equivalent to linear temporal logic (*LTL* for short), star-free expressions and counter-free Büchi automata (cf. for instance [7]). More interestingly, *LTL* and its alternatives serve as specification languages in model checking for real world applications [3, 22, 31]. The last few years there is also an increasing interest in establishing *FO* logic and its equivalent objects in the quantitative framework. This is motivated by the need to create model checking tools which incorporate quantitative features. In [14], the aforementioned equivalence was established in the weighted setup of arbitrary bounded lattices. Recently, in [26] (cf. also [24]), we introduced a weighted *FO* logic, a weighted *LTL*, ω-star-free series

*Research of the first author has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) - Research Funding Program: Heracleitus II. Investing in knowledge society through the European Social Fund.

†Department of Mathematics, Aristotle University of Thessaloniki, 54124 Thessaloniki, Greece. E-mail: elemandr@gmail.com

‡E-mail: grahonis@math.auth.gr

and counter-free weighted Büchi automata over the max-plus semiring with discounting and investigated fragments of them satisfying an expressive equivalence. The convergence of infinite sums over nonnegative real numbers was ensured by the existence of discounting parameters.

In this paper, we consider a weighted *FO* logic, a weighted *LTL*, ω-star-free series and counter-free weighted Büchi automata over idempotent, zero-divisor free and totally commutative complete semirings. We show that there are suitable fragments of our objects so that the classes of infinitary series, derived by them, coincide. Our results can be proved for series over finite words as well, though we skip any technical detail.

The structure of our paper is as follows. Except of this introductory section, in Section 2 we recall the notion of totally commutative complete semirings and present notations used in the paper. The underlying structure for all weighted objects considered in the paper will be an arbitrary idempotent, zero-divisor free and totally commutative complete semiring.

In Section 3 we introduce the weighted *LTL* and define the semantics of *LTL* formulas interpreted as infinitary series. We consider a fragment of our *LTL* namely the fragment of U-nesting formulas. We should note that a quantitative *LTL* over De Morgan algebras was introduced for the first time in [21].

In Section 4 we consider the weighted *FO* logic which is in fact the one induced by the weighted *MSO* logic of [8, 9]. Its semantics is interpreted by infinitary series as induced by the semantics of the corresponding weighted *MSO* logic of [13]. We consider the fragment of weakly quantified *FO* logic formulas and in our first main result, in Section 5, we show that every series which is definable by a U-nesting *LTL* formula is definable also by a weakly quantified *FO* logic sentence.

In Section 6 we deal with star-free and ω-star-free series. We recall that the class of star-free languages over an alphabet A is the smallest class of languages over A which contains $\emptyset$, the singleton $\{a\}$ for every $a \in A$, and which is closed under finite union, complementation and concatenation. Furthermore, the class of ω-star-free languages over A is the closure of the empty set under the operations of union, complement and concatenation with star-free languages on the left (cf. for instance [7, 23, 27, 29]). It is worth noting that the application of the star-operation (whenever it is permitted) to star-free languages is implemented by the other operations. However, in the setup of series (over semirings) the complement operation is not "too strong". Therefore, we defined the class ω-star-free series as the least class of infinitary series generated by the monomials (over A and our semiring) by applying finitely many times the operations of sum, Hadamard product, complement, Cauchy product, and iteration and ω-iteration restricted to series of the form $\sum_{a\in A}(k_a)_a$ where, for every $a \in A$, k_a is an element of our semiring. The second main result of the paper, in Section 7, states that the class of definable series by weakly quantified *FO* logic sentences is contained in the class of ω-star-free series.

In Section 8 we introduce counter-free weighted automata and counter-free weighted Büchi automata and investigate closure properties of the classes of their behaviors. We define a fragment of the class of series accepted by counter-free weighted Büchi automata, namely the class of almost simple ω-counter-free series

and we show, in Section 9, that this contains the class of ω-star-free series.

Finally, in Section 10 we show that the class of almost simple ω-counter-free series is contained in the class of series which are definable by U-nesting LTL formulas. In fact this last inclusion concludes the coincidence of the classes of series definable by U-nesting formulas of the weighted LTL and weakly quantified FO logic sentences, ω-star-free series and almost simple ω-counter-free series. In the Conclusion we refer to some interesting problems for further research. A preliminary version of this paper appeared in [25].

2 Preliminaries

Let A be an alphabet, i.e., a finite nonempty set. As usually, we denote by A^* the set of all finite words over A and $A^+ = A^* \setminus \{\varepsilon\}$, where ε is the empty word. The set of all infinite sequences with elements in A, i.e., the set of all infinite words over A, is denoted by A^ω. A finite word $w = a_0 \dots a_{n-1}$, where $a_0, \dots, a_{n-1} \in A$ $(n \geq 1)$, is written also as $w = w(0) \dots w(n-1)$ where $w(i) = a_i$ for every $0 \leq i \leq n-1$. For every $0 \leq i \leq n-1$, we denote by $w_{<i}$ (resp. $w_{\leq i}$) the prefix $w(0) \dots w(i-1)$ (resp. $w(0) \dots w(i)$) of w and by $w_{>i}$ (resp. $w_{\geq i}$) the suffix $w(i+1) \dots w(n-1)$ (resp. $w(i) \dots w(n-1)$) of w. For every infinite word $w = a_0 a_1 \dots$ which is written also as $w = w(0)w(1) \dots$, the words $w_{<i}, w_{\leq i}, w_{>i}, w_{\geq i}$ are defined in the same way, with the suffixes $w_{>i}, w_{\geq i}$ being infinite words.

Throughout the paper A will denote an alphabet.

A *semiring* $(K, +, \cdot, 0, 1)$ consists of a set K, two binary operations $+$ and $\cdot$ and two constant elements 0 and 1 such that $\langle K, +, 0\rangle$ is a commutative monoid, $\langle K, \cdot, 1\rangle$ is a monoid, multiplication distributes over addition, and $0 \cdot k = k \cdot 0 = 0$ for every $k \in K$. The semiring is denoted simply by K if the operations and the constant elements are understood.

The semiring K is called *commutative* if $k \cdot k' = k' \cdot k$ for every $k, k' \in K$. It is called *additively idempotent* (or simply *idempotent*), if $k + k = k$ for every $k \in K$. Moreover, the semiring K is *zero-sum free* (resp. *zero-divisor free*) if $k + k' = 0$ implies $k = k' = 0$ (resp. $k \cdot k' = 0$ implies $k = 0$ or $k' = 0$) for every $k, k' \in K$. It is well known that every idempotent semiring is necessarily zero-sum free (cf. [1]).

Next, assume that the semiring K is equipped, for every index set I, with infinitary sum operations $\sum_I : K^I \to K$, such that for every family $(k_i \mid i \in I)$ of elements of K and $k \in K$ we have

$$\sum_{i \in \emptyset} k_i = 0, \quad \sum_{i \in \{j\}} k_i = k_j, \quad \sum_{i \in \{j,l\}} k_i = k_j + k_l \text{ for } j \neq l,$$

$$\sum_{j \in J} \Big(\sum_{i \in I_j} k_i \Big) = \sum_{i \in I} k_i, \text{ if } \bigcup_{j \in J} I_j = I \text{ and } I_j \cap I_{j'} = \emptyset \text{ for } j \neq j',$$

$$\sum_{i \in I} (k \cdot k_i) = k \cdot \Big(\sum_{i \in I} k_i \Big), \quad \sum_{i \in I} (k_i \cdot k) = \Big(\sum_{i \in I} k_i \Big) \cdot k.$$

Then the semiring K together with the operations $\sum_I$ is called *complete* [15, 19].

A complete semiring is said to be *totally complete* [18], if it is endowed with a countably infinite product operation satisfying for every sequence $(k_i \mid i \geq 0)$ of elements of K the subsequent conditions:

$$\prod_{i\geq 0} 1 = 1, \quad \prod_{i\geq 0} k_i = \prod_{i\geq 0} k'_i$$

$$k_0 \cdot \prod_{i\geq 0} k_{i+1} = \prod_{i\geq 0} k_i, \quad \prod_{j\geq 1} \sum_{i\in I_j} k_i = \sum_{(i_1,i_2,\ldots)\in I_1\times I_2\times\ldots} \prod_{j\geq 1} k_{i_j},$$

where in the second equation $k'_0 = k_0 \cdot \ldots \cdot k_{n_1}, k'_1 = k_{n_1+1} \cdot \ldots \cdot k_{n_2}, \ldots$ for an increasing sequence $0 < n_1 < n_2 < \ldots$, and in the last equation $I_1, I_2, \ldots$ are arbitrary index sets.

Furthermore, we will call a totally complete semiring K *totally commutative complete* if it satisfies the statement:

$$\prod_{i\geq 0} (k_i \cdot k'_i) = \left(\prod_{i\geq 0} k_i\right) \cdot \left(\prod_{i\geq 0} k'_i\right).$$

Obviously a totally commutative complete semiring is commutative. For our theory, we shall also need that a totally commutative complete semiring K satisfies the property

$$k \neq 0 \implies \prod_{i\geq 0} k \neq 0$$

for every $k \in K$. Therefore in the sequel, by abusing terminology, when we refer to totally commutative complete semirings we assume that they additionally satisfy the above property.

Example 1. The following semirings are totally commutative complete, and all but the second one are idempotent. Moreover, by excluding the arbitrary completely distributive complete lattices, the remaining ones are zero-divisor free.

- the *boolean semiring* $\mathbb{B} = (\{\mathbf{0}, \mathbf{1}\}, +, \cdot, \mathbf{0}, \mathbf{1})$,
- the semiring $(\mathbb{N} \cup \{\infty\}, +, \cdot, 0, 1)$ of *extended natural numbers* [17],
- the *arctical semiring* or *max-plus semiring* $(\mathbb{R}_+ \cup \{\pm\infty\}, \max, +, -\infty, 0)$,
- each completely distributive complete lattice (cf. [2]) with the operations supremum and infimum, in particular each complete chain [20].

Lemma 1. *Let K be an idempotent totally complete semiring and I an index set of size at most continuum. Then, the following statements hold.*

(i) [10, Chap. 5, Lm. 7.3] $\sum_I 1 = 1$.

(ii) $\sum_I k = k$ *for every* $k \in K$.

(iii) $\sum_{i\in I} k_i = \sum_{\substack{k\in K \\ \exists i\in I, k_i=k}} k$ *for every family* $(k_i)_{i\in I}$ *in* K.

Proof. (ii) By (i) and distributivity we get $\sum_I k = k \cdot \sum_I 1 = k \cdot 1 = k$.

(iii) For every $k \in K$ we let $I_k = \{i \in I \mid k_i = k\}$. Then we get

$$\sum_{i\in I} k_i = \sum_{\substack{k\in K \\ \exists i\in I, k_i=k}} \sum_{I_k} k = \sum_{\substack{k\in K \\ \exists i\in I, k_i=k}} k$$

where the second equality follows by (ii). □

In the rest of the paper K will denote a totally commutative complete, idempotent and zero-divisor free semiring.

Let Q be a set. A *formal power series* (or simply *series*) *over Q and K* is a mapping $s : Q \to K$. For every $v \in Q$ we write (s, v) for the value $s(v)$ and refer to it as the *coefficient of s on v*. The *support of s* is the set $\mathrm{supp}(s) = \{v \in Q \mid (s, v) \neq 0\}$. The *constant series* $\widetilde{k}$ $(k \in K\ )$ is defined, for every $v \in Q$, by $\left(\widetilde{k}, v\right) = k$. The *characteristic series* 1_P *of a set* $P \subseteq Q$ is given by $(1_P, v) = 1$ if $v \in P$, and $(1_P, v) = 0$ otherwise. We denote by $K\langle\langle Q\rangle\rangle$ the class of all series over Q and K.

Let $s, r \in K\langle\langle Q\rangle\rangle$ and $k \in K$. The *sum* $s + r$, the *scalar products* ks and sk as well as the *Hadamard product* $s \odot r$ are defined elementwise by $(s + r, v) = (s, v) + (r, v)$, $(ks, v) = k \cdot (s, v)$, $(sk, v) = (s, v) \cdot k$, and $(s \odot r, v) = (s, v) \cdot (r, v)$ for every $v \in Q$. Abusing notations, if $P \subseteq Q$, then we shall identify the restriction $s|_P$ of s on P with the series $s \odot 1_P$. Moreover, if $\mathrm{supp}\,(s) \subseteq P$, sometimes in the sequel we shall identify $s|_P$ with s. It is a folklore result that the structure $\left(K\langle\langle Q\rangle\rangle, +, \odot, \widetilde{0}, \widetilde{1}\right)$ is a commutative semiring. In our paper, we work with the semirings $K\langle\langle A^*\rangle\rangle$ and $K\langle\langle A^\omega\rangle\rangle$ of finitary and infinitary series over A and K, respectively.

Let B be another alphabet and $h : A^* \to B^*$ be a nondeleting homomorphism, i.e., $h(a) \neq \varepsilon$ for each $a \in A$. Then h can be extended to a mapping $h : A^\omega \to B^\omega$ by letting $h(w) = (h(w(i)))_{i\geq 0}$ for every $w \in A^\omega$. Moreover, h is extended to a mapping $h : K\langle\langle A^*\rangle\rangle \to K\langle\langle B^*\rangle\rangle$ as follows. For every $s \in K\langle\langle A^*\rangle\rangle$ the series $h(s) \in K\langle\langle B^*\rangle\rangle$ is given by $(h(s), u) = \sum_{w\in h^{-1}(u)}(s, w)$ for every $u \in B^*$. Since K is complete, h is also extended to a mapping $h : K\langle\langle A^\omega\rangle\rangle \to K\langle\langle B^\omega\rangle\rangle$ which is defined for every series $s \in K\langle\langle A^\omega\rangle\rangle$ by $(h(s), u) = \sum_{w\in h^{-1}(u)}(s, w)$ for every $u \in B^\omega$. If $r \in K\langle\langle B^*\rangle\rangle$ (resp. $r \in K\langle\langle B^\omega\rangle\rangle$), then the series $h^{-1}(r) \in K\langle\langle A^*\rangle\rangle$ (resp. $h^{-1}(r) \in K\langle\langle A^\omega\rangle\rangle$) is determined by $(h^{-1}(r), w) = (r, h(w))$ for every $w \in A^*$ (resp. $w \in A^\omega$).

3 Weighted linear temporal logic

For every letter $a \in A$ we consider a proposition p_a and we let $AP = \{p_a \mid a \in A\}$. As usually, for every $p \in AP$ we identify $\neg\neg p$ with p.

Definition 1. *The syntax of formulas of the* weighted linear temporal logic *(weighted LTL for short)* over A and K *is given by the grammar*

$$\varphi ::= k \mid p_a \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \bigcirc\varphi \mid \varphi U \varphi \mid \Box\varphi$$

where $k \in K$ and $p_a \in AP$.

We denote by $LTL(K, A)$ the set of all such weighted LTL formulas φ. We represent the semantics $\|\varphi\|$ of formulas $\varphi \in LTL(K, A)$ as infinitary series in $K\langle\langle A^\omega \rangle\rangle$.

Definition 2. *Let $\varphi \in LTL(K, A)$. The* semantics *of φ is a series $\|\varphi\| \in K\langle\langle A^\omega \rangle\rangle$ which is defined inductively as follows. For every $w \in A^\omega$ we set*

- $(\|k\|, w) = k$,
- $(\|p_a\|, w) = \begin{cases} 1 & \text{if } w(0) = a \\ 0 & \text{otherwise} \end{cases}$,
- $(\|\neg\varphi\|, w) = \begin{cases} 1 & \text{if } (\|\varphi\|, w) = 0 \\ 0 & \text{otherwise} \end{cases}$,
- $(\|\varphi \vee \psi\|, w) = (\|\varphi\|, w) + (\|\psi\|, w)$,
- $(\|\varphi \wedge \psi\|, w) = (\|\varphi\|, w) \cdot (\|\psi\|, w)$,
- $(\|\bigcirc\varphi\|, w) = (\|\varphi\|, w_{\geq 1})$,
- $(\|\varphi U \psi\|, w) = \sum_{i \geq 0} \left(\left(\prod_{0 \leq j < i} (\|\varphi\|, w_{\geq j}) \right) \cdot (\|\psi\|, w_{\geq i}) \right)$,
- $(\|\Box\varphi\|, w) = \prod_{i \geq 0} (\|\varphi\|, w_{\geq i})$.

The *eventually* operator is defined as in the classical LTL, i.e., by $\Diamond\varphi := 1U\varphi$, hence we have $(\|\Diamond\varphi\|, w) = \sum_{i \geq 0} (\|\varphi\|, w_{\geq i})$ for every $w \in A^\omega$.

The syntactic boolean fragment $bLTL(K, A)$ of $LTL(K, A)$ is given by the grammar

$$\varphi ::= 0 \mid 1 \mid p_a \mid \neg\varphi \mid \varphi \vee \varphi \mid \bigcirc\varphi \mid \varphi U \varphi$$

where $p_a \in AP$. For every formula $\varphi \in bLTL(K, A)$ it is easily obtained, by structural induction on φ and using idempotency, that $\|\varphi\|$ gets only values in $\{0, 1\}$. By identifying 0 with $\mathbf{0}$ and 1 with $\mathbf{1}$ it is trivially concluded that $\|\varphi\|$ coincides with

the semantics in the boolean semiring $\mathbb{B}$. The *conjunction* and *always operators* are defined, respectively, by the macros $\varphi \underline{\wedge} \psi := \neg(\neg\varphi \vee \neg\psi)$ and $\underline{\Box}\varphi := \neg\Diamond\neg\varphi$. Clearly, the application of the operators $\underline{\wedge}$ and $\underline{\Box}$ in $bLTL(K,A)$ formulas φ, ψ coincides semantically with the application of the classical operators $\wedge$ and $\Box$ in φ, ψ considered as classical formulas.

We aim to define a further fragment of $LTL(K,A)$. For this we need some preliminary matter. More precisely, an *atomic-step formula* is an $LTL(K,A)$ formula of the form $\bigvee_{a\in A}(k_a \wedge p_a)$ where $k_a \in K$ and $p_a \in AP$ for every $a \in A$. An *LTL-step formula* is an $LTL(K,A)$ formula of the form $\bigvee_{1\leq i\leq n}(k_i \wedge \varphi_i)$ where $k_i \in K$ and $\varphi_i \in bLTL(K,A)$ for every $1 \leq i \leq n$. We shall denote by $stLTL(K,A)$ the class of *LTL*-step formulas over A and K. Furthermore, we shall denote by $abLTL(K,A)$ the class of *almost boolean LTL* formulas over A and K, i.e., formulas of the form $\bigwedge_{1\leq i\leq n}\varphi_i$ with $\varphi_i \in bLTL(K,A)$ or $\varphi_i = \bigvee_{a\in A}(k_a \wedge p_a)$, for every $1 \leq i \leq n$.

Definition 3. *The fragment $ULTL(K,A)$ of* U-nesting *LTL* formulas over A and K *is the least class of formulas in $LTL(K,A)$ which is defined inductively in the following way.*

- $k \in ULTL(K,A)$ *for every* $k \in K$.
- $abLTL(K,A) \subseteq ULTL(K,A)$.
- *If* $\varphi \in ULTL(K,A)$, *then* $\neg\varphi \in ULTL(K,A)$.
- *If* $\varphi, \psi \in ULTL(K,A)$, *then* $\varphi \wedge \psi, \varphi \vee \psi \in ULTL(K,A)$.
- *If* $\varphi \in ULTL(K,A)$, *then* $\bigcirc\varphi \in ULTL(K,A)$.
- *If* $\varphi \in bLTL(K,A)$ *or* φ *is an atomic-step formula, then* $\Box\varphi \in ULTL(K,A)$.
- *If* $\varphi \in abLTL(K,A)$ *and* $\psi \in ULTL(K,A)$, *then* $\varphi U \psi \in ULTL(K,A)$.

A series $r \in K\langle\langle A^\omega\rangle\rangle$ is called *ω-ULTL-definable* if there is a formula $\varphi \in ULTL(K,A)$ such that $r = \|\varphi\|$. We shall denote by $\omega\text{-}ULTL(K,A)$ the class of ω-*ULTL*-definable series over A and K.

4 Weighted first-order logic

In this section, we define the weighted first-order logic (weighted *FO* logic, for short) and consider a syntactic fragment of it. We aim to show that the class of semantics of sentences in this fragment contains the class $\omega\text{-}ULTL(K,A)$.

Definition 4. *The syntax of formulas of the* weighted *FO* logic over A and K *is given by the grammar*

$$\varphi ::= k \mid P_a(x) \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \varphi \wedge \varphi \mid \exists x \,.\, \varphi \mid \forall x \,.\, \varphi$$

where $k \in K$ *and* $a \in A$.

We shall denote by $FO(K,A)$ the set of all weighted FO logic formulas over A and K. In order to define the semantics of $FO(K,A)$ formulas, we recall the notions of extended alphabet and valid assignment (cf. for instance [30]). Let $\mathcal{V}$ be a finite set of first-order variables. For an infinite word $w \in A^\omega$ we let $dom(w) = \omega$. A $(\mathcal{V}, w)$-*assignment* σ is a mapping associating variables from $\mathcal{V}$ to elements of ω. For every $x \in \mathcal{V}$ and $i \in \omega$, we denote by $\sigma[x \to i]$ the $(\mathcal{V}, w)$-assignment which associates i to x and acts as σ on $\mathcal{V} \setminus \{x\}$. We encode pairs (w, σ) for every $w \in A^\omega$ and $(\mathcal{V}, w)$-assignment σ, by using the extended alphabet $A_\mathcal{V} = A \times \{0,1\}^\mathcal{V}$. Each word in $A_\mathcal{V}^\omega$ can be considered as a pair (w, σ) where w is the projection over A and σ is the projection over $\{0,1\}^\mathcal{V}$. Then, σ is called a *valid* $(\mathcal{V}, w)$-assignment whenever for every $x \in \mathcal{V}$ the x-row contains exactly one 1. In this case, we identify σ with the $(\mathcal{V}, w)$-assignment so that for every first-order variable $x \in \mathcal{V}$, $\sigma(x)$ is the position of the 1 on the x-row. It is well-known (cf. [7]) that the set $\mathcal{N}_\mathcal{V} = \{(w, \sigma) \mid w \in A^\omega,\ \sigma \text{ is a valid } (\mathcal{V}, w)\text{-assignment}\}$ is an ω-star-free language over $A_\mathcal{V}$. The set $free(\varphi)$ of free variables in a formula $\varphi \in FO(K,A)$ is defined as usual.

Definition 5. *Let $\varphi \in FO(K,A)$ and $\mathcal{V}$ be a finite set of variables with $free(\varphi) \subseteq \mathcal{V}$. The semantics of φ is a series $\|\varphi\|_\mathcal{V} \in K\langle\langle A_\mathcal{V}^\omega \rangle\rangle$. Consider an element $(w, \sigma) \in A_\mathcal{V}^\omega$. If σ is not a valid assignment, then we put $(\|\varphi\|_\mathcal{V}, (w, \sigma)) = 0$. Otherwise, we inductively define $(\|\varphi\|_\mathcal{V}, (w, \sigma)) \in K$ as follows.*

- $(\|k\|_\mathcal{V}, (w, \sigma)) = k$,
- $(\|P_a(x)\|_\mathcal{V}, (w, \sigma)) = \begin{cases} 1 & \text{if } w(\sigma(x)) = a \\ 0 & \text{otherwise} \end{cases}$,
- $(\|x \leq y\|_\mathcal{V}, (w, \sigma)) = \begin{cases} 1 & \text{if } \sigma(x) \leq \sigma(y) \\ 0 & \text{otherwise} \end{cases}$,
- $(\|\neg\varphi\|_\mathcal{V}, (w, \sigma)) = \begin{cases} 1 & \text{if } (\|\varphi\|_\mathcal{V}, (w, \sigma)) = 0 \\ 0 & \text{otherwise} \end{cases}$,
- $(\|\varphi \vee \psi\|_\mathcal{V}, (w, \sigma)) = (\|\varphi\|_\mathcal{V}, (w, \sigma)) + (\|\psi\|_\mathcal{V}, (w, \sigma))$,
- $(\|\varphi \wedge \psi\|_\mathcal{V}, (w, \sigma)) = (\|\varphi\|_\mathcal{V}, (w, \sigma)) \cdot (\|\psi\|_\mathcal{V}, (w, \sigma))$,
- $(\|\exists x \centerdot \varphi\|_\mathcal{V}, (w, \sigma)) = \sum_{i \geq 0} \left(\|\varphi\|_{\mathcal{V} \cup \{x\}}, (w, \sigma[x \to i])\right)$,
- $(\|\forall x \centerdot \varphi\|_\mathcal{V}, (w, \sigma)) = \prod_{i \geq 0} \left(\|\varphi\|_{\mathcal{V} \cup \{x\}}, (w, \sigma[x \to i])\right)$.

If $\mathcal{V} = free(\varphi)$, then we simply write $\|\varphi\|$ for $\|\varphi\|_{free(\varphi)}$. Moreover, by Prop. 5 in [13], it holds

$$(\|\varphi\|_\mathcal{V}, (w, \sigma)) = \left(\|\varphi\|, \left(w, \sigma|_{free(\varphi)}\right)\right)$$

for every $(w, \sigma) \in \mathcal{N}_\mathcal{V}$.

The syntactic boolean fragment $bFO(K,A)$ of $FO(K,A)$ is defined by the grammar

$$\varphi ::= 0 \mid 1 \mid P_a(x) \mid x \leq y \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x \,.\, \varphi.$$

For every formula $\varphi \in bFO(K,A)$ it is easily obtained, by structural induction on φ and using idempotency, that $\|\varphi\|$ gets only values in $\{0,1\}$. By identifying 0 with $\mathbf{0}$ and 1 with $\mathbf{1}$ it is trivially concluded that $\|\varphi\|$ coincides with the semantics in the boolean semiring $\mathbb{B}$. The *conjunction* and *universal quantification* are defined, respectively, by the macros $\varphi \underline{\wedge} \psi := \neg(\neg\varphi \vee \neg\psi)$ and $\underline{\forall} x \,.\, \varphi := \neg\exists x \,.\, \neg\varphi$. Clearly, the application of the operators $\underline{\wedge}$ and $\underline{\forall}$ in $bFO(K,A)$ formulas φ, ψ coincides semantically with the application of the classical operators $\wedge$ and $\forall$ in φ, ψ considered as classical formulas.

Next, we define a fragment of our logic. For this, we recall the notion of an FO-step formula from [4]. More precisely, a formula $\varphi \in FO(K,A)$ is an *FO-step formula* if $\varphi = \bigvee_{1 \leq i \leq n} (k_i \wedge \varphi_i)$ with $\varphi_i \in bFO(K,A)$ and $k_i \in K$ for every $1 \leq i \leq n$. Moreover, a formula $\varphi \in FO(K,A)$ is called a *letter-step formula* whenever $\varphi = \bigvee_{a \in A} (k_a \wedge P_a(x))$ with $k_a \in K$ for every $a \in A$. We shall need also the following macros:

- $first(x) := \forall y \,.\, x \leq y$,
- $x = y := x \leq y \wedge y \leq x$,
- $x < y := x \leq y \wedge \neg(x = y)$,
- $z \leq x < y := z \leq x \wedge x < y$,
- $\varphi \rightarrow \psi := \neg\varphi \vee (\varphi \wedge \psi)$.

Definition 6. *A formula $\varphi \in FO(K,A)$ will be called* weakly quantified *if whenever φ contains a subformula of the form $\forall x \,.\, \psi$, then ψ is either a boolean or a letter-step formula with free variable x or a formula of the form $y \leq x \rightarrow \psi'$ or $z \leq x < y \rightarrow \psi'$ where ψ' is a letter-step formula with free variable x.*

We denote by $WQFO(K,A)$ the set of all weakly quantified $FO(K,A)$ formulas over A and K. A series $s \in K\langle\langle A^\omega \rangle\rangle$ is called *ω-$wqFO$-definable* if there is a sentence $\varphi \in WQFO(K,A)$ such that $s = \|\varphi\|$. We write $\omega\text{-}wqFO(K,A)$ for the class of ω-$wqFO$-definable series in $K\langle\langle A^\omega \rangle\rangle$.

5 ω-$ULTL$-definable series are ω-$wqFO$-definable

In this section we show that every ω-$ULTL$-definable series over A and K is also ω-$wqFO$-definable. For this, we will prove that for every $\varphi \in ULTL(K,A)$ there exists a sentence $\varphi' \in WQFO(K,A)$ such that $\|\varphi\| = \|\varphi'\|$, using the subsequent technical results.

Lemma 2. *Let $\varphi \in ULTL(K,A)$ such that there exists $\varphi'(y) \in WQFO(K,A)$ with*
$(\|\varphi'(y)\|, (w, [y \rightarrow i])) = (\|\varphi\|, w_{\geq i})$ for every $w \in A^\omega, i \geq 0$.
Then $(\|\neg\varphi'(y)\|, (w, [y \rightarrow i])) = (\|\neg\varphi\|, w_{\geq i})$ for every $w \in A^\omega, i \geq 0$.

Lemma 3. *Let $\varphi, \psi \in ULTL(K,A)$ such that there exist $\varphi'(y), \psi'(x) \in WQFO(K,A)$ with $(\|\varphi'(y)\|, (w,[y \to i])) = (\|\varphi\|, w_{\geq i})$ and $(\|\psi'(x)\|, (w,[x \to i])) = (\|\psi\|, w_{\geq i})$ for every $w \in A^\omega, i \geq 0$. Then, there exist $\xi_1(x), \xi_2(x) \in WQFO(K,A)$ with*

$$(\|\xi_1(x)\|, (w,[x \to i])) = (\|\varphi \wedge \psi\|, w_{\geq i})$$

and

$$(\|\xi_2(x)\|, (w,[x \to i])) = (\|\varphi \vee \psi\|, w_{\geq i})$$

for every $w \in A^\omega, i \geq 0$.

Proof. Without any loss, we assume that the variable x does not occur in φ' (otherwise we apply a renaming). We replace every occurrence of y with x in φ', and we let $\xi_1(x) = \varphi'(x) \wedge \psi'(x)$ and $\xi_2(x) = \varphi'(x) \vee \psi'(x)$ which trivially satisfy our claim. □

Lemma 4. *Let $\varphi \in K \cup abLTL(K,A)$. Then, there exists $\varphi'(x) \in WQFO(K,A)$ such that $(\|\varphi'(x)\|, (w,[x \to i])) = (\|\varphi\|, w_{\geq i})$ for every $w \in A^\omega, i \geq 0$.*

Proof. Let $\varphi = k \in K$. Then we set $\varphi'(x) = k$. Next, let $\varphi \in abLTL(K,A)$, i.e., $\varphi = \bigwedge_{1 \leq j \leq n} \psi_j$ with $\psi_j \in bLTL(K,A)$ or $\psi_j = \bigvee_{a \in A}(k_a \wedge p_a)$, for every $1 \leq j \leq n$. If $\psi_j \in bLTL(K,A)$, then it is well-known that there exists a formula $\psi'_j(x_j) \in bFO(K,A)$ with one free variable x_j, such that $(\|\psi_j\|, w_{\geq i}) = \left(\left\|\psi'_j(x_j)\right\|, (w,[x_j \to i])\right)$ for every $w \in A^\omega, i \geq 0$. Without any loss, we can assume that the variable x_j $(1 \leq j \leq n)$ does not occur in any ψ'_k (whenever $\psi'_k \in bLTL(K,A)$) with $k \neq j$ (if this is not the case, then we apply a renaming of variables). Therefore, we can replace x_j in ψ'_j with a new variable x. In case $\psi_j = \bigvee_{a \in A}(k_a \wedge p_a)$ we consider the $WQFO(K,A)$ letter-step formula $\psi'_j(x) = \bigvee_{a \in A}(k_a \wedge P_a(x))$. Now it is a routine matter to show that the $WQFO(K,A)$ formula $\varphi'(x) = \bigwedge_{1 \leq j \leq n} \psi'_j(x)$ satisfies our claim. □

Lemma 5. *Let $\varphi \in ULTL(K,A)$ such that there exists a formula $\varphi'(y) \in WQFO(K,A)$ with $(\|\varphi'(y)\|, (w,[y \to i])) = (\|\varphi\|, w_{\geq i})$ for every $w \in A^\omega, i \geq 0$. Then, there exists a $WQFO(K,A)$ formula $\psi(x)$ such that $(\|\psi(x)\|, (w,[x \to i])) = (\|\bigcirc\varphi\|, w_{\geq i})$ for every $w \in A^\omega, i \geq 0$.*

Proof. We let $\psi(x) = \exists y.(y = x + 1 \wedge \varphi'(y))$ and we have

$$\begin{aligned}
(\|\psi(x)\|, (w, [x \to i])) &= (\|\exists y.(y = x + 1 \wedge \varphi'(y))\|, (w, [x \to i])) \\
&= \sum_{j \geq 0} (\|y = x + 1 \wedge \varphi'(y)\|, (w, [x \to i, y \to j])) \\
&= (\|y = x + 1 \wedge \varphi'(y)\|, (w, [x \to i, y \to i + 1])) \\
&\quad + \sum_{j \geq 0, j \neq i+1} (\|y = x + 1 \wedge \varphi'(y)\|, (w, [x \to i, y \to j])) \\
&= (\|y = x + 1 \wedge \varphi'(y)\|, (w, [x \to i, y \to i + 1])) \\
&= (\|\varphi'(y)\|, (w, [y \to i + 1])) \\
&= (\|\varphi\|, w_{\geq i+1}) = (\|\bigcirc\varphi\|, w_{\geq i}).
\end{aligned}$$

for every $w \in A^\omega, i \geq 0$, where the fourth equality holds by Lemma 1(ii). □

Lemma 6. *Let $\varphi \in bLTL(K, A)$ or φ be an atomic-step formula. Then, there exists $\psi(y) \in WQFO(K, A)$ such that $(\|\psi(y)\|, (w, [y \to i])) = (\|\Box\varphi\|, w_{\geq i})$ for every $w \in A^\omega, i \geq 0$.*

Proof. If $\varphi \in bLTL(K, A)$, then $\Box\varphi \in bLTL(K, A)$, and thus there exists a formula $\psi(x) \in bFO(K, A)$ with one free variable x, such that $(\|\psi(x)\|, (w, [x \to i])) = (\|\Box\varphi\|, w_{\geq i})$ for every $w \in A^\omega, i \geq 0$. If $\varphi = \bigvee_{a \in A}(k_a \wedge p_a)$, then we consider the $WQFO(K, A)$ letter-step formula $\varphi'(x) = \bigvee_{a \in A}(k_a \wedge P_a(x))$. We also consider the $WQFO(K, A)$ formula $\psi(y) = \forall x.(y \leq x \to \varphi'(x))$. Then, for every $w \in A^\omega, i \geq 0$ we have

$$\begin{aligned}
(\|\psi(y)\|, (w, [y \to i])) &= \prod_{j \geq 0} (\|y \leq x \to \varphi'(x)\|, (w, [y \to i, x \to j])) \\
&= \prod_{j \geq i} (\|y \leq x \wedge \varphi'(x)\|, (w, [y \to i, x \to j])) \\
&= \prod_{j \geq i} (\|\varphi'(x)\|, (w, [x \to j])) \\
&= \prod_{j \geq i} (\|\varphi\|, w_{\geq j}) \\
&= (\|\Box\varphi\|, w_{\geq i})
\end{aligned}$$

where the fourth equality holds by Lemma 4. □

Lemma 7. *Let $\varphi \in abLTL(K, A)$ and $\psi \in ULTL(K, A)$ such that there exists $\psi'(y) \in WQFO(K, A)$ with $(\|\psi'(y)\|, (w, [y \to i])) = (\|\psi\|, w_{\geq i})$ for every $w \in A^\omega, i \geq 0$. Then, there exists $\xi(z) \in WQFO(K, A)$ such that $(\|\xi(z)\|, (w, [z \to i])) = (\|\varphi U \psi\|, w_{\geq i})$ for every $w \in A^\omega, i \geq 0$.*

Proof. Let $\varphi = \bigwedge_{1\leq l\leq m} \varphi_l$. Then, by the proof of Lemma 4, there exists a formula $\varphi'(x) = \bigwedge_{1\leq l\leq m} \varphi'_l(x)$ where for every $1 \leq l \leq m$, $\varphi'_l(x) \in bFO(K,A)$ or it is a letter-step formula with $(\|\varphi'_l(x)\|(w,[x\to i])) = (\|\varphi_l\|, w_{\geq i})$ for every $w \in A^\omega, i \geq 0$. Moreover, we have $(\|\varphi'(x)\|, (w,[x\to i])) = (\|\varphi\|, w_{\geq i})$ for every $w \in A^\omega, i \geq 0$. We consider the $FO(K,A)$ formula $\xi'(z) = \exists y.(\forall x.((z \leq x < y) \to \varphi'(x)) \wedge (z \leq y) \wedge \psi'(y))$. For every $w \in A^\omega, i \geq 0$ we compute

$$
\begin{aligned}
&(\|\xi'(z)\|, (w,[z\to i])) \\
&= \sum_{j\geq 0} (\|\forall x.((z \leq x < y) \to \varphi'(x)) \wedge (z \leq y) \wedge \psi'(y)\|, (w,[z\to i, y\to j])) \\
&= \sum_{j\geq 0} (\|\forall x.((z \leq x < y) \to \varphi'(x)) \wedge \psi'(y)\|, (w,[z\to i, y\to i+j])) \\
&= \sum_{j\geq 0} \left(\left(\prod_{0\leq k<j} (\|\varphi'(x)\|, (w,[x\to i+k])) \right) \cdot (\|\psi'(y)\|, (w,[y\to i+j])) \right) \\
&= \sum_{j\geq 0} \left(\left(\prod_{0\leq k<j} (\|\varphi\|, w_{\geq i+k}) \right) \cdot (\|\psi\|, w_{\geq i+j}) \right) \\
&= (\|\varphi U \psi\|, w_{\geq i}).
\end{aligned}
$$

Now, we consider the formula

$$\xi(z) = \exists y.\left(\bigwedge_{1\leq l\leq m} (\forall x.((z \leq x < y) \to \varphi'_l(x))) \wedge (z \leq y) \wedge \psi'(y) \right)$$

and for every $w \in A^\omega, i \geq 0$ we get $(\|\xi(z)\|, (w,[z\to i])) = (\|\xi'(z)\|, (w,[z\to i])) = (\|\varphi U\psi\|, w_{\geq i})$. Since $\xi(z) \in WQFO(K,A)$, we conclude our proof. □

Lemma 8. *For every $ULTL(K,A)$ formula φ we can construct a $WQFO(K,A)$ formula $\varphi'(x)$ such that $(\|\varphi'(x)\|, (w,[x\to i])) = (\|\varphi\|, w_{\geq i})$ for every $w \in A^\omega, i \geq 0$.*

Proof. We use Lemmas 2, 3, 4, 5, 6, and 7. □

Proposition 1. *For every $\varphi \in ULTL(K,A)$ we can construct a $WQFO(K,A)$ sentence φ' with $\|\varphi'\| = \|\varphi\|$.*

Proof. Let $\varphi \in ULTL(K,A)$. By the previous lemma, there exists a $WQFO(K,A)$ formula $\psi(x)$ such that $(\|\psi(x)\|, (w,[x\to i])) = (\|\varphi\|, w_{\geq i})$, for every $w \in A^\omega, i \geq 0$. We consider the $WQFO(K,A)$ sentence $\varphi' = \exists x.(first(x) \wedge \psi(x))$ and we get $(\|\varphi'\|, w) = (\|\psi(x)\|, (w,[x\to 0])) = (\|\varphi\|, w)$ for every $w \in A^\omega$, i.e., $\|\varphi'\| = \|\varphi\|$, as required. □

By the above proposition, we get the main result of this section.

Theorem 1. *ω-ULTL$(K,A) \subseteq \omega$-wqFO(K,A).*

The result of the next corollary, which is trivially obtained by the constructive proofs of this section's lemmas and propositions, in fact generalizes the corresponding result that relates boolean *LTL* and *FO* logic.

Corollary 1. *For every $\varphi \in ULTL(K,A)$ we can construct a $WQFO(K,A)$ sentence φ', that uses at most three different names of variables, such that $\|\varphi'\| = \|\varphi\|$.*

6 Star-free series

In this section, we introduce the notions of star-free and ω-star-free series over A and K. Let $L \subseteq A^*$ (resp. $L \subseteq A^\omega$). As usually, we denote by 1_L the characteristic series of L. If L is a singleton, i.e., $L = \{w\}$, then we simply write 1_w for $1_{\{w\}}$. Furthermore, we simply denote by k_L the series $k1_L$ for $k \in K$. The *monomials over A and K* are series of the form $(k_a)_a$ for $a \in A$ and $k_a \in K$. For simplicity, we shall consider also the series of the form k_ε with $k \in K$ as monomials. A series $s \in K\langle\langle A^*\rangle\rangle$ is called a *letter-step series* if $s = \sum_{a\in A}(k_a)_a$ where $k_a \in K$ for every $a \in A$. The *complement $\overline{s}$ of a series s* is given by $(\overline{s}, w) = 1$ if $(s,w) = 0$, and $(\overline{s}, w) = 0$ otherwise. Let $r, s \in K\langle\langle A^*\rangle\rangle$. The (*Cauchy*) *product of r and s* is the series $r \cdot s \in K\langle\langle A^*\rangle\rangle$ defined for every $w \in A^*$ by

$(r \cdot s, w) = \sum\{(r,u)\cdot(s,v) \mid u, v \in A^*, w = uv\}$.

The nth-iteration $r^n \in K\langle\langle A^*\rangle\rangle$ $(n \geq 0)$ of a series $r \in K\langle\langle A^*\rangle\rangle$ is defined inductively by

$r^0 = 1_\varepsilon$ and $r^{n+1} = r \cdot r^n$ for $n \geq 0$.

Then, we have $(r^n, w) = \sum\left\{\prod_{1\leq i\leq n}(r,u_i) \mid u_i \in A^*, w = u_1 \dots u_n\right\}$ for every $w \in A^*$. A series $r \in K\langle\langle A^*\rangle\rangle$ is called *proper* if $(r,\varepsilon) = 0$. If r is proper, then for every $w \in A^*$ and $n > |w|$ we have $(r^n, w) = 0$. The *iteration $r^+ \in K\langle\langle A^*\rangle\rangle$ of a proper series $r \in K\langle\langle A^*\rangle\rangle$* is defined by $r^+ = \sum_{n>0} r^n$. Thus, for every $w \in A^+$ we have $(r^+, w) = \sum_{1\leq n\leq|w|}(r^n, w)$ and $(r^+, \varepsilon) = 0$.

Definition 7. *The class of* star-free series over A and K, *denoted by $SF(K,A)$, is the least class of series containing the monomials (over A and K) and being closed under sum, Hadamard product, complement, Cauchy product, and iteration restricted to letter-step series.*

Next, let $r \in K\langle\langle A^*\rangle\rangle$ be a finitary and $s \in K\langle\langle A^\omega\rangle\rangle$ an infinitary series. Then, the *Cauchy product of r and s* is the infinitary series $r \cdot s \in K\langle\langle A^\omega\rangle\rangle$ defined for every $w \in A^\omega$ by

$(r \cdot s, w) = \sum\{(r,u)\cdot(s,v) \mid u \in A^*, v \in A^\omega, w = uv\}$ [1].

[1] Since the semiring K is idempotent (resp. By Lemma 1(ii)), the notation of the sum in the definition of Cauchy product of two finitary series (resp. of a finitary and an infinitary series), is consistent with the standard definition.

The *ω-iteration of a proper finitary series* $r \in K\langle\langle A^*\rangle\rangle$ is the infinitary series $r^\omega \in K\langle\langle A^\omega\rangle\rangle$ which is defined by

$(r^\omega, w) = \sum \left\{ \prod_{i\geq 1}(r, u_i) \mid u_i \in A^*, w = u_1 u_2 \dots \right\}$

for every $w \in A^\omega$.

Example 2. Let $r = \sum_{a\in A}(k_a)_a \in K\langle\langle A^*\rangle\rangle$ be a letter-step series. We will show that $(r^+)^+ = r^+$. Moreover, for every $w \in A^\omega$ we have $(r^\omega, w) = \prod_{i\geq 0}(r, w(i))$.

Let $w = w(0)\dots w(n-1) \in A^+$. Then

$$\begin{aligned}(r^+, w) &= \sum \left\{ \prod_{1\leq j\leq k} (r, u_j) \mid w = u_1 \dots u_k, 1 \leq k \leq n \right\} \\ &= \prod_{0\leq j\leq n-1} (r, w(j)).\end{aligned}$$

Furthermore, we get

$$\begin{aligned}&\left((r^+)^+, w\right) \\ &= \sum \left\{ \prod_{1\leq j\leq k} (r^+, u_j) \mid w = u_1 \dots u_k, 1 \leq k \leq n \right\} \\ &= \sum \left\{ \prod_{1\leq j\leq k} \left(\prod_{0\leq i_j\leq |u_j|-1} (r, u_j(i_j)) \right) \mid w = u_1 \dots u_k, 1 \leq k \leq n \right\} \\ &= \prod_{0\leq j\leq n-1} (r, w(j)) = (r^+, w).\end{aligned}$$

Similarly, we can show that $(r^\omega, w) = \prod_{i\geq 0}(r, w(i))$, for every $w \in A^\omega$.

Definition 8. *The class of* ω-star-free series over A and K, *denoted by* ω-$SF(K, A)$, *is the least class of infinitary series generated by the monomials (over A and K) by applying finitely many times the operations of sum, Hadamard product, complement, Cauchy product, iteration restricted to letter-step series, and ω-iteration restricted to letter-step series.*

The next result is trivially proved by Definitions 7, 8 and standard arguments.

Lemma 9. *Let $r \in SF(K, A)$ (resp. $r \in \omega$-$SF(K, A)$) and $B \subseteq A$. Then $r|_{B^*} \in SF(K, B)$ (resp. $r|_{B^\omega} \in \omega$-$SF(K, B)$).*

In the sequel, we state properties of the classes $SF(K, A)$ and ω-$SF(K, A)$. More precisely, we prove a splitting lemma and the closure of the classes under inverse strict alphabetic epimorphisms and bijections.

Lemma 10. *If $r \in SF(K, A)$ (resp. $r \in \omega$-$SF(K, A)$) and $k \in K$, then $kr \in SF(K, A)$ (resp. $kr \in \omega$-$SF(K, A)$).*

Proof. We have $kr = k_\varepsilon \cdot r$, hence we get the proof of our claim. □

Lemma 11. *Let $L, L' \subseteq A^*$ and $K, K' \subseteq A^\omega$. Then*

- $1_{L \cup L'} = 1_L + 1_{L'}, \quad 1_{K \cup K'} = 1_K + 1_{K'}$
- $1_{L \cap L'} = 1_L \odot 1_{L'}, \quad 1_{K \cap K'} = 1_K \odot 1_{K'}$
- $1_{LL'} = 1_L \cdot 1_{L'}, \quad 1_{LK} = 1_L \cdot 1_K$
- $1_{L^+} = (1_L)^+ \quad$ *whenever* $\varepsilon \notin L$
- $1_{L^\omega} = (1_L)^\omega \quad$ *whenever* $\varepsilon \notin L$.

Proof. We use standard arguments and the idempotency property of the semiring K. In particular, for the last statement we use Lemma 1(i). □

The two subsequent results are shown by induction on the structure of star-free (resp. ω-star-free) languages and series using Lemma 11.

Lemma 12. *For every $L \subseteq A^*$ the following statements are equivalent.*

(i) *L is a star-free language.*

(ii) $1_L \in SF(K, A)$.

Lemma 13. *For every $L \subseteq A^\omega$ the following statements are equivalent.*

(i) *L is an ω-star-free language.*

(ii) $1_L \in \omega\text{-}SF(K, A)$.

Since for every $L \subseteq A^*$ (resp. $L \subseteq A^\omega$) and $k \in K$ we have $k_L = k_\varepsilon \cdot 1_L$, by Lemmas 12 and 13, we get Lemma 14 below.

Lemma 14. *Let $L \subseteq A^*$ (resp. $L \subseteq A^\omega$) and $k \in K$. If L is star-free (resp. ω-star-free), then $k_L \in SF(K, A)$ (resp. $k_L \in \omega\text{-}SF(K, A)$).*

Lemma 15. *If $s \in SF(K, A)$ (resp. $s \in \omega\text{-}SF(K, A)$), then* $\mathrm{supp}(s)$ *is a star-free language (resp. an ω-star-free) language over A.*

Proof. Using standard arguments, we state the proof by induction on the structure of s. □

Lemma 16.

(i) *Let $L \subseteq A^*$ be a star-free language and $B, \Gamma \subseteq A$ with $B \cap \Gamma = \emptyset$. Then $1_L|_{B^*\Gamma B^*} = \sum_{1 \le i \le n} \left(1_{M_i} \cdot \left(1_{\gamma_i} \cdot 1_{M_i'}\right)\right)$ where for every $1 \le i \le n$, $M_i, M_i' \subseteq B^*$ are star-free languages, and $\gamma_i \in \Gamma$.*

(ii) Let $L \subseteq A^\omega$ be an ω-star-free language and $B, \Gamma \subseteq A$ with $B \cap \Gamma = \emptyset$. Then $1_L|_{B^*\Gamma B^\omega} = \sum_{1\leq i\leq n} \left(1_{M_i} \cdot \left(1_{\gamma_i} \cdot 1_{M_i'}\right)\right)$ where for every $1 \leq i \leq n$, $M_i \subseteq B^*$ is star-free, $M_i' \subseteq B^\omega$ is ω-star-free, and $\gamma_i \in \Gamma$.

Proof. We prove only (ii); Statement (i) is shown with the same arguments. By the splitting lemma for ω-star-free languages (cf. Lm. 3.2. in [7]), we get $L \cap B^*\Gamma B^\omega = \bigcup_{1\leq i\leq n} M_i\gamma_i M_i'$ where for every $1 \leq i \leq n$, $M_i \subseteq B^*$ is star-free, $\gamma_i \in \Gamma$, and $M_i' \subseteq B^\omega$ is ω-star-free. Since $1_L|_{B^*\Gamma B^\omega} = 1_{L\cap B^*\Gamma B^\omega}$, we complete our proof using Lemma 11. □

Proposition 2 (Splitting lemma for finitary series)**.** *Let $s \in SF(K, A)$ and $B, \Gamma \subseteq A$ with $B \cap \Gamma = \emptyset$. Then $s|_{B^*\Gamma B^*} = \sum_{1\leq i\leq n} \left(s_1^{(i)} \cdot \left(s_2^{(i)} \cdot s_3^{(i)}\right)\right)$ where for every $1 \leq i \leq n$, $s_1^{(i)}, s_3^{(i)} \in SF(K, B)$ and $s_2^{(i)} = (k_i)_{\gamma_i}$ with $\gamma_i \in \Gamma, k_i \in K$.*

Proof. We use induction on the structure of s. Let $s = (k_a)_a$, $a \in A$, be a monomial. Then, if $a \in \Gamma$, we have $s|_{B^*\Gamma B^*} = 1_\varepsilon \cdot ((k_a)_a \cdot 1_\varepsilon)$, otherwise $s|_{B^*\Gamma B^*} = 1_\emptyset \cdot \left((k_\gamma)_\gamma \cdot 1_\emptyset\right)$ for an arbitrary $\gamma \in \Gamma$. If $s = k_\varepsilon$, then again $s|_{B^*\Gamma B^*} = 1_\emptyset \cdot \left((k_\gamma)_\gamma \cdot 1_\emptyset\right)$ for an arbitrary $\gamma \in \Gamma$.

Let $s, r \in SF(K, A)$ satisfying the induction hypothesis. This means that $s|_{B^*\Gamma B^*} = \sum_{1\leq i\leq n} \left(s_1^{(i)} \cdot \left(s_2^{(i)} \cdot s_3^{(i)}\right)\right)$ and $r|_{B^*\Gamma B^*} = \sum_{1\leq j\leq m} \left(r_1^{(j)} \cdot \left(r_2^{(j)} \cdot r_3^{(j)}\right)\right)$ where for every $1 \leq i \leq n$ and $1 \leq j \leq m$, we have $s_1^{(i)}, s_3^{(i)}, r_1^{(j)}, r_3^{(j)} \in SF(K, B)$, $s_2^{(i)} = (k_i)_{\gamma_i}$, $r_2^{(j)} = (l_j)_{\gamma_j'}$, $\gamma_i, \gamma_j' \in \Gamma$, $k_i, l_j \in K$. Obviously, $(s + r)|_{B^*\Gamma B^*}$ has the required form.

Next let $w \in B^*\Gamma B^*$ and $0 \leq k \leq |w| - 1$ with $w(k) \in \Gamma$. Then $w_{<k}, w_{>k} \in B^*$ and we have

$$\begin{aligned}(s|_{B^*\Gamma B^*}, w) &= \left(\sum_{1\leq i\leq n} \left(s_1^{(i)} \cdot \left(s_2^{(i)} \cdot s_3^{(i)}\right)\right), w\right) \\ &= \sum_{1\leq i\leq n} \left(s_1^{(i)} \cdot \left(s_2^{(i)} \cdot s_3^{(i)}\right), w\right) \\ &= \sum_{1\leq i\leq n} \left(\left(s_1^{(i)}, w_{<k}\right) \cdot \left(s_2^{(i)}, w(k)\right) \cdot \left(s_3^{(i)}, w_{>k}\right)\right)\end{aligned}$$

where the third equality holds since for every $1 \leq i \leq n$ and every decomposition $w = u_1u_2u_3$ with $u_2 \neq w(k)$ we have $\left(s_2^{(i)}, u_2\right) = 0$.
Similarly

$$\begin{aligned}(r|_{B^*\Gamma B^*}, w) &= \left(\sum_{1\leq j\leq m} \left(r_1^{(j)} \cdot \left(r_2^{(j)} \cdot r_3^{(j)}\right)\right), w\right) \\ &= \sum_{1\leq j\leq m} \left(\left(r_1^{(j)}, w_{<k}\right) \cdot \left(r_2^{(j)}, w(k)\right) \cdot \left(r_3^{(j)}, w_{>k}\right)\right).\end{aligned}$$

Hence,

$$\begin{aligned}
&((s \odot r)\,|_{B^*\Gamma B^*}, w) = (s|_{B^*\Gamma B^*}, w) \cdot (r|_{B^*\Gamma B^*}, w) \\
&= \sum_{1\le i\le n} \left(\left(s_1^{(i)}, w_{<k}\right) \cdot \left(s_2^{(i)}, w(k)\right) \cdot \left(s_3^{(i)}, w_{>k}\right)\right) \\
&\cdot \sum_{1\le j\le m} \left(\left(r_1^{(j)}, w_{<k}\right) \cdot \left(r_2^{(j)}, w(k)\right) \cdot \left(r_3^{(j)}, w_{>k}\right)\right) \\
&= \sum_{\substack{1\le i\le n \\ 1\le j\le m}} \left(\left(s_1^{(i)} \odot r_1^{(j)}, w_{<k}\right) \cdot \left(\left(s_2^{(i)} \odot r_2^{(j)}, w(k)\right) \cdot \left(s_3^{(i)} \odot r_3^{(j)}, w_{>k}\right)\right)\right) \\
&= \left(\sum_{\substack{1\le i\le n \\ 1\le j\le m}} \left(\left(s_1^{(i)} \odot r_1^{(j)}\right) \cdot \left(\left(s_2^{(i)} \odot r_2^{(j)}\right) \cdot \left(s_3^{(i)} \odot r_3^{(j)}\right)\right)\right), w\right).
\end{aligned}$$

Since $s_1^{(i)} \odot r_1^{(j)}$, $s_3^{(i)} \odot r_3^{(j)} \in SF(K, B)$, and $s_2^{(i)} \odot r_2^{(j)} = (k_i \cdot l_j)_{\gamma_i}$ if $\gamma_i = \gamma'_j$, and $s_2^{(i)} \odot r_2^{(j)} = 0_\gamma$ for an arbitrary $\gamma \in \Gamma$ otherwise, our claim is true for the Hadamard product.

Furthermore,

$$\begin{aligned}
((s \cdot r)\,|_{B^*\Gamma B^*}, w) = &\sum \{(s|_{B^*\Gamma B^*}, u) \cdot (r, v) \mid u \in B^*\Gamma B^*, v \in B^*, w = uv\} \\
&+ \sum \{(s, u) \cdot (r|_{B^*\Gamma B^*}, v) \mid u \in B^*, v \in B^*\Gamma B^*, w = uv\}
\end{aligned}$$

with

$$\begin{aligned}
&\sum \{(s|_{B^*\Gamma B^*}, u) \cdot (r, v) \mid u \in B^*\Gamma B^*, v \in B^*, w = uv\} \\
&= \sum \left\{ \left(\sum_{1\le i\le n} \left(s_1^{(i)} \cdot \left(s_2^{(i)} \cdot s_3^{(i)}\right)\right), u\right) \cdot (r, v) \mid u \in B^*\Gamma B^*, v \in B^*, w = uv \right\} \\
&= \sum \left\{ \left(\sum_{1\le i\le n} \left(s_1^{(i)} \cdot \left(s_2^{(i)} \cdot s_3^{(i)}\right)\right), u\right) \cdot (r|_{B^*}, v) \mid u, v \in A^*, w = uv \right\} \\
&= \left(\left(\sum_{1\le i\le n} \left(s_1^{(i)} \cdot \left(s_2^{(i)} \cdot s_3^{(i)}\right)\right)\right) \cdot r|_{B^*}, w\right) \\
&= \left(\sum_{1\le i\le n} \left(\left(s_1^{(i)} \cdot \left(s_2^{(i)} \cdot s_3^{(i)}\right)\right) \cdot r|_{B^*}\right), w\right) \\
&= \left(\sum_{1\le i\le n} \left(s_1^{(i)} \cdot \left(s_2^{(i)} \cdot \left(s_3^{(i)} \cdot r|_{B^*}\right)\right)\right), w\right)
\end{aligned}$$

where $r|_{B^*} = r \odot 1_{B^*} \in SF(K, B)$, and the fourth equality holds since the Cauchy product distributes over the sum of series. Similarly

$$\begin{aligned}
&\sum \{(s,u) \cdot (r|_{B^*\Gamma B^*}, v) \mid u \in B^*, v \in B^*\Gamma B^*, w = uv\} \\
&= \sum \left\{ (s,u) \cdot \left(\sum_{1 \le j \le m} \left(r_1^{(j)} \cdot \left(r_2^{(j)} \cdot r_3^{(j)} \right) \right), v \right) \mid u \in B^*, v \in B^*\Gamma B^*, w = uv \right\} \\
&= \sum \left\{ (s|_{B^*}, u) \cdot \left(\sum_{1 \le j \le m} \left(r_1^{(j)} \cdot \left(r_2^{(j)} \cdot r_3^{(j)} \right) \right), v \right) \mid u, v \in A^*, w = uv \right\} \\
&= \left(s|_{B^*} \cdot \sum_{1 \le j \le m} \left(r_1^{(j)} \cdot \left(r_2^{(j)} \cdot r_3^{(j)} \right) \right), w \right) \\
&= \left(\sum_{1 \le j \le m} \left(s|_{B^*} \cdot \left(r_1^{(j)} \cdot \left(r_2^{(j)} \cdot r_3^{(j)} \right) \right) \right), w \right) \\
&= \left(\sum_{1 \le j \le m} \left(\left(s|_{B^*} \cdot r_1^{(j)} \right) \cdot \left(r_2^{(j)} \cdot r_3^{(j)} \right) \right), w \right).
\end{aligned}$$

Thus,

$$\begin{aligned}
((s \cdot r)|_{B^*\Gamma B^*}, w) = &\left(\sum_{1 \le i \le n} \left(s_1^{(i)} \cdot \left(s_2^{(i)} \cdot \left(s_3^{(i)} \cdot r|_{B^*} \right) \right) \right), w \right) \\
&+ \left(\sum_{1 \le j \le m} \left(\left(s|_{B^*} \cdot r_1^{(j)} \right) \cdot \left(r_2^{(j)} \cdot r_3^{(j)} \right) \right), w \right).
\end{aligned}$$

Therefore, the series $(s \cdot r)|_{B^*\Gamma B^*}$ has the required form.

Now, let s be a letter-step series. Then, $s|_{B^*\Gamma B^*} = s|_\Gamma = \sum_{\gamma \in \Gamma} (k_\gamma)_\gamma$. Let $w \in \mathrm{supp}(s^+) \cap B^*\Gamma B^*$, which implies that there is an index $0 \le k \le |w| - 1$ such

that $w_{<k}, w_{>k} \in B^*$ and $w(k) \in \Gamma$. Then

$$\begin{aligned}
&\left(\left(s^{+}\right)|_{B^*\Gamma B^*}, w\right) \\
&= \sum \{(s^m|_{B^*\Gamma B^*}, w) \mid 1 \leq m \leq |w|\} = \left(s^{|w|}|_{B^*\Gamma B^*}, w\right) \\
&= \prod_{0\leq j\leq |w|-1} (s, w(j)) \\
&= \left(\prod_{0\leq j\leq k-1} (s, w(j))\right) \cdot (s, w(k)) \cdot \left(\prod_{k<j\leq |w|-1} (s, w(j))\right) \\
&= \left(\left(s|_B\right)^{+} \cdot \left(s|_\Gamma \cdot \left(s|_B\right)^{+}\right), w\right) \\
&= \left(\sum_{\gamma\in\Gamma} \left(\left(s|_B\right)^{+} \cdot \left(\left(k_\gamma\right)_\gamma \cdot \left(s|_B\right)^{+}\right)\right), w\right)
\end{aligned}$$

and this concludes the induction for letter-step series.

Finally, let $s \in SF(K, A)$. Then $\overline{s} = 1_{\overline{\mathrm{supp}(s)}}$. Since $\mathrm{supp}(s)$ is a star-free language, we get that $\overline{\mathrm{supp}(s)}$ is also star-free. Hence, by Lemma 16(i) we conclude our proof. □

Proposition 3 (Splitting lemma for infinitary series)**.** *Let $s \in \omega\text{-}SF(K, A)$ and $B, \Gamma \subseteq A$ with $B \cap \Gamma = \emptyset$. Then $s|_{B^*\Gamma B^\omega} = \sum_{1\leq i\leq n} \left(s_1^{(i)} \cdot \left(s_2^{(i)} \cdot s_3^{(i)}\right)\right)$ where for every $1 \leq i \leq n$, $s_1^{(i)} \in SF(K, B)$, $s_3^{(i)} \in \omega\text{-}SF(K, B)$, and $s_2^{(i)} = (k_i)_{\gamma_i}$ with $\gamma_i \in \Gamma, k_i \in K$.*

Proof. Taking into account the definition of ω-star-free series, firstly we embed the proof of Lemma 2. Furthermore, we use arguments of that proof as follows. For the operations of sum and Hadamard product we let $s, r \in \omega\text{-}SF(K, A)$, and for Cauchy product we let $s \in SF(K, A)$ and $r \in \omega\text{-}SF(K, A)$. For the complement operation, we let $s \in \omega\text{-}SF(K, A)$ and we use the corresponding argument for ω-star-free languages and Lemma 16(ii). Finally, let s be a letter-step series. Then, $s|_{B^*\Gamma B^*} = s|_\Gamma = \sum_{\gamma\in\Gamma} (k_\gamma)_\gamma$. Let $w \in \mathrm{supp}(s^\omega) \cap B^*\Gamma B^\omega$, i.e., there exists an

index $k \geq 0$ such that $w_{<k} \in B^*, w_{>k} \in B^\omega$, and $w(k) \in \Gamma$. Then we get

$$\begin{aligned}
&((s^\omega)\,|_{B^*\Gamma B^\omega}, w)\\
&= \sum \left\{ \prod_{i\geq 1} (s, u_i) \mid u_i \in A^*, w = u_1 u_2 \dots \right\}\\
&= \prod_{j\geq 0} (s, w(j))\\
&= \left(\prod_{0\leq j\leq k-1} (s, w(j)) \right) \cdot (s, w(k)) \cdot \left(\prod_{j>k} (s, w(j)) \right)\\
&= \left((s|_B)^+ \cdot (s|_\Gamma \cdot (s|_B)^\omega), w \right)\\
&= \left(\sum_{\gamma\in\Gamma} \left((s|_B)^+ \cdot \left((k_\gamma)_\gamma \cdot (s|_B)^\omega \right) \right), w \right)
\end{aligned}$$

i.e.,

$$(s^\omega)\,|_{B^*\Gamma B^\omega} = \sum_{\gamma\in\Gamma} \left((s|_B)^+ \cdot \left((k_\gamma)_\gamma \cdot (s|_B)^\omega \right) \right)$$

and this completes our proof. □

Proposition 4. *Let A, B be two alphabets and $h : A \to B$ a bijection. Then $s \in SF(K, A)$ (resp. $s \in \omega\text{-}SF(K, A)$) implies that $h(s) \in SF(K, B)$ (resp. $h(s) \in \omega\text{-}SF(K, B)$).*

Proof. There is an one-to-one correspondence between the words of A^* and B^* (resp. the words of A^ω and B^ω) derived by h. Then, we can easily state our proof by induction on the structure of star-free (resp. ω-star-free) series. □

Proposition 5. *Let A, B be alphabets and $h : A \to B$ a strict alphabetic epimorphism. Then $s \in SF(K, B)$ (resp. $s \in \omega\text{-}SF(K, B)$) implies that $h^{-1}(s) \in SF(K, A)$ (resp. $h^{-1}(s) \in \omega\text{-}SF(K, A)$).*

Proof. We prove our claim by induction on the structure of star-free (resp. ω-star-free) series. Let $s = (k_b)_b$ be a monomial over B and K. Then, $h^{-1}(s)$ is a letter-step series and thus a star-free series over A and K. If $s = k_\varepsilon$, then $h^{-1}(s) = k_\varepsilon$ since h is strict. Next let $s_1, s_2 \in SF(K, B)$ (resp. $s_1, s_2 \in \omega\text{-}SF(K, B)$) such that $h^{-1}(s_1), h^{-1}(s_2) \in SF(K, A)$ (resp. $h^{-1}(s_1), h^{-1}(s_2) \in \omega\text{-}SF(K, A)$). Trivially $h^{-1}(s_1 \odot s_2) = h^{-1}(s_1) \odot h^{-1}(s_2)$ and $h^{-1}(s_1 + s_2) = h^{-1}(s_1) + h^{-1}(s_2)$.

Furthermore, for every $w \in A^*$ we have

$$\begin{aligned}\left(h^{-1}(s_1 \cdot s_2), w\right) &= (s_1 \cdot s_2, h(w)) \\ &= \sum \{(s_1, u_1) \cdot (s_2, u_2) \mid u_1, u_2 \in B^*, u_1 u_2 = h(w)\} \\ &= \sum \{(s_1, h(w_1)) \cdot (s_2, h(w_2)) \mid w_1, w_2 \in A^*, w_1 w_2 = w\} \\ &= \sum \left\{\left(h^{-1}(s_1), w_1\right) \cdot \left(h^{-1}(s_2), w_2\right) \mid w_1, w_2 \in A^*, w_1 w_2 = w\right\} \\ &= \left(h^{-1}(s_1) \cdot h^{-1}(s_2), w\right)\end{aligned}$$

where the fourth equality holds since h is strict alphabetic. Hence $h^{-1}(s_1 \cdot s_2) = h^{-1}(s_1) \cdot h^{-1}(s_2)$. If $s_1 \in SF(K, B)$, $s_2 \in \omega\text{-}SF(K, B)$, and $w \in A^\omega$, then we use the same as above argument, where we write $u_2 \in B^\omega$ and $w_2 \in A^\omega$.

Assume now that s is a letter-step series over B and K. Then, the series $h^{-1}(s)$ is a letter-step series over A and K, hence $h^{-1}(s) \in SF(K, A)$. For every $w \in A^+$ we get

$$\begin{aligned}(h^{-1}(s^+), w) = (s^+, h(w)) &= \prod_{0 \le j \le |w|-1} (s, h(w)(j)) \\ &= \prod_{0 \le j \le |w|-1} (s, h(w(j))) = \prod_{0 \le j \le |w|-1} (h^{-1}(s), w(j)) \\ &= ((h^{-1}(s))^+, w),\end{aligned}$$

i.e., $h^{-1}(s^+) = (h^{-1}(s))^+ \in SF(K, A)$.

Next, let $s \in SF(K, B)$. Then, $\overline{s} = 1_{\overline{\mathrm{supp}(s)}}$ and $\overline{\mathrm{supp}(s)}$ is, by Lemma 12, a star-free language over B. Moreover, the language $h^{-1}\left(\overline{\mathrm{supp}(s)}\right) \subseteq A^*$ is star-free (cf. for instance [28]) hence, the series $h^{-1}(\overline{s}) = h^{-1}\left(1_{\overline{\mathrm{supp}(s)}}\right) = 1_{h^{-1}\left(\overline{\mathrm{supp}(s)}\right)}$ is star-free by Lemma 12. The case $s \in \omega\text{-}SF(K, B)$ is treated similarly.

Finally, assume that s is a letter-step series over B and K. Then, $h^{-1}(s)$ is a letter-step series over A and K. Moreover, for every $w \in A^\omega$ we have

$$\begin{aligned}(h^{-1}(s^\omega), w) = (s^\omega, h(w)) &= \prod_{j \ge 0} (s, h(w)(j)) \\ &= \prod_{j \ge 0} (s, h(w(j))) = \prod_{j \ge 0} (h^{-1}(s), w(j)) \\ &= ((h^{-1}(s))^\omega, w),\end{aligned}$$

i.e., $h^{-1}(s^\omega) = \left(h^{-1}(s)\right)^\omega \in \omega\text{-}SF(A, K)$, and our proof is completed. □

7 ω-$wqFO$-definable series are ω-star-free

In the sequel, we show that every ω-$wqFO$-definable series over A and K is an ω-star-free series, i.e., $\omega\text{-}wqFO(K, A) \subseteq \omega\text{-}SF(K, A)$. For this, we use induction

on the structure of $WQFO(K,A)$ formulas. We shall need the following auxiliary result.

Lemma 17. *Let $\varphi \in FO(K,A)$ and $\mathcal{V}$ be a finite set of first-order variables containing $free(\varphi)$. If $\|\varphi\|$ is an ω-star-free series, then $\|\varphi\|_{\mathcal{V}}$ is an ω-star-free series.*

Proof. Let $\|\varphi\|$ be an ω-star-free series and $h : A_{\mathcal{V}} \to A_{free(\varphi)}$ the strict alphabetic epimorphism erasing the x-row for every $x \in \mathcal{V} \setminus free(\varphi)$. It holds $\|\varphi\|_{\mathcal{V}} = h^{-1}(\|\varphi\|) \odot 1_{\mathcal{N}_{\mathcal{V}}}$. Then by Proposition 5 we get that $h^{-1}(\|\varphi\|) \in \omega\text{-}SF(K, A_{\mathcal{V}})$, and thus $\|\varphi\|_{\mathcal{V}} \in \omega\text{-}SF(K, A_{\mathcal{V}})$, as wanted. □

Lemma 18. *Let $\varphi \in FO(K,A)$ be an atomic formula. Then, $\|\varphi\|$ is an ω-star-free series.*

Proof. If $\varphi = k \in K$, then $\|\varphi\| = k_{A^\omega}$. Next, if $\varphi = P_a(x)$ or $x \leq y$, then φ is a boolean first-order formula, hence $\mathcal{L}(\varphi)$ is an ω-star-free language and $\|\varphi\| = 1_{\mathcal{L}(\varphi)}$ is an ω-star-free series. □

Lemma 19. *Let $\varphi \in FO(K,A)$ such that $\|\varphi\|$ is an ω-star-free series. Then, $\|\neg\varphi\|$ is also an ω-star-free series.*

Proof. By definition, we have $\|\neg\varphi\| = \overline{\|\varphi\|}$. □

Lemma 20. *Let $\varphi, \psi \in FO(K,A)$. If $\|\varphi\|, \|\psi\|$ are ω-star-free series, then $\|\varphi \wedge \psi\|, \|\varphi \vee \psi\|$ are ω-star-free series.*

Proof. Let $\mathcal{V} = free(\varphi) \cup free(\psi)$. We have $\|\varphi \wedge \psi\| = \|\varphi\|_{\mathcal{V}} \odot \|\psi\|_{\mathcal{V}}$ and $\|\varphi \vee \psi\| = \|\varphi\|_{\mathcal{V}} + \|\psi\|_{\mathcal{V}}$, hence our claim follows by definition of ω-star-free series and Lemma 17. □

Lemma 21. *Let $\varphi \in FO(K,A)$ such that $\|\varphi\|$ is an ω-star-free series. Then, $\|\exists x.\varphi\|$ is also an ω-star-free series.*

Proof. Let $\mathcal{W} = free(\varphi) \cup \{x\}$ and $\mathcal{V} = free(\exists x.\varphi) = \mathcal{W} \setminus \{x\}$. We define two subalphabets B, Γ of $A_{\mathcal{W}}$ by letting $B = \{(a,f) \in A_{\mathcal{W}} \mid f(x) = 0\}$ and $\Gamma = \{(a,f) \in A_{\mathcal{W}} \mid f(x) = 1\}$. Since $\|\varphi\|_{\mathcal{W}} \in \omega\text{-}SF(K, A_{\mathcal{W}})$ (by Lemma 17, in case $x \notin free(\varphi)$), by Proposition 3 we get

$$\|\varphi\|_{\mathcal{W}} |_{B^*\Gamma B^\omega} = \sum_{1 \leq i \leq n} \left(s_1^{(i)} \cdot \left(s_2^{(i)} \cdot s_3^{(i)} \right) \right)$$

with $s_1^{(i)} \in SF(K,B)$, $s_3^{(i)} \in \omega\text{-}SF(K,B)$, and $s_2^{(i)} = (k_i)_{\gamma_i}$, where $k_i \in K$, $\gamma_i \in \Gamma$ for every $1 \leq i \leq n$. We show that

$$\|\exists x.\varphi\| = \left(\sum_{1 \leq i \leq n} \left(h|_B \left(s_1^{(i)} \right) \cdot \left((k_i)_{h(\gamma_i)} \cdot h|_B \left(s_3^{(i)} \right) \right) \right) \right) \odot 1_{\mathcal{N}_{\mathcal{V}}}$$

where $h : A_{\mathcal{W}} \to A_{\mathcal{V}}$ is the strict alphabetic epimorphism assigning $(a, f|_{\mathcal{V}})$ to (a, f) for every $(a, f) \in A_{\mathcal{W}}$.

Let $(w, \sigma) \in \mathcal{N}_{\mathcal{V}}$. Then we have

$$
\begin{aligned}
&(\|\exists x.\varphi\|, (w,\sigma)) \\
&= \sum_{j\geq 0} (\|\varphi\|_{\mathcal{W}}, (w, \sigma[x \to j])) \\
&= \sum_{j\geq 0} (\|\varphi\|_{\mathcal{W}}|_{B^*\Gamma B^\omega}, (w, \sigma[x \to j])) \\
&= \sum_{j\geq 0} \left(\sum_{1\leq i\leq n} \left(s_1^{(i)} \cdot \left(s_2^{(i)} \cdot s_3^{(i)} \right) \right), (w, \sigma[x \to j]) \right) \\
&= \sum_{j\geq 0} \left(\sum_{1\leq i\leq n} \left(\begin{array}{r} \left(s_1^{(i)}, (w, \sigma[x \to j])_{<j} \right) \cdot \left(s_2^{(i)}, (w, \sigma[x \to j])(j) \right) \\ \cdot \left(s_3^{(i)}, (w, \sigma[x \to j])_{>j} \right) \end{array} \right) \right) \\
&= \sum_{1\leq i\leq n} \left(\sum_{j\geq 0} \left(\begin{array}{r} \left(s_1^{(i)}, (w, \sigma[x \to j])_{<j} \right) \cdot \left(s_2^{(i)}, (w, \sigma[x \to j])(j) \right) \\ \cdot \left(s_3^{(i)}, (w, \sigma[x \to j])_{>j} \right) \end{array} \right) \right) \\
&= \sum_{1\leq i\leq n} \left(\sum_{j\geq 0} \left(\begin{array}{r} \left(h|_B\left(s_1^{(i)} \right), (w,\sigma)_{<j} \right) \cdot \left((k_i)_{h(\gamma_i)}, (w,\sigma)(j) \right) \\ \cdot \left(h|_B\left(s_3^{(i)} \right), (w,\sigma)_{>j} \right) \end{array} \right) \right) \\
&= \sum_{1\leq i\leq n} \left(h|_B\left(s_1^{(i)} \right) \cdot \left((k_i)_{h(\gamma_i)} \cdot h|_B\left(s_3^{(i)} \right) \right), (w,\sigma) \right)
\end{aligned}
$$

where the sixth equality holds since $h\left((k_i)_{\gamma_i}\right) = (k_i)_{h(\gamma_i)}$ and $h|_B : B \to A_{\mathcal{V}}$ is a bijection. On the other hand, for every $(w, \sigma) \in A_{\mathcal{V}}^{\omega} \setminus \mathcal{N}_{\mathcal{V}}$ we have

$$
\left(\sum_{1\leq i\leq n} \left(h|_B\left(s_1^{(i)} \right) \cdot \left((k_i)_{h(\gamma_i)} \cdot h|_B\left(s_3^{(i)} \right) \right) \right) \odot 1_{\mathcal{N}_{\mathcal{V}}}, (w,\sigma) \right) = 0.
$$

Hence, $\|\exists x.\varphi\| = \left(\sum_{1\leq i\leq n} \left(h|_B\left(s_1^{(i)} \right) \cdot \left((k_i)_{h(\gamma_i)} \cdot h|_B\left(s_3^{(i)} \right) \right) \right) \right) \odot 1_{\mathcal{N}_{\mathcal{V}}}$. By Proposition 4, for every $1 \leq i \leq n$, we get that $h|_B\left(s_1^i\right) \in SF(K, A_{\mathcal{V}})$, $h|_B\left(s_3^{(i)}\right) \in \omega\text{-}SF(K, A_{\mathcal{V}})$. Therefore $\|\exists x.\varphi\|$ is an ω-star-free series. □

Lemma 22. *Let $\varphi \in FO(K, A)$ be a boolean, or a letter-step formula with free variable x, or $\varphi = (y \leq x) \to \psi$, or $\varphi = (y \leq x < z) \to \psi$ where ψ is a letter-step formula with free variable x. Then, $\|\forall x.\varphi\|$ is an ω-star-free series.*

Proof. If $\varphi \in bFO(K, A)$, then $\forall x.\varphi \in bFO(K, A)$, hence the language $\mathcal{L}(\forall x.\varphi)$ is ω-star-free and the series $\|\forall x.\varphi\| = 1_{\mathcal{L}(\forall x.\varphi)}$ is ω-star-free.

Next, assume that $\varphi = \bigvee_{a\in A}(k_a \wedge P_a(x))$ is a letter-step formula with $k_a \in K$ for every $a \in A$. We consider the letter-step series $r = \sum_{a\in A}(k_a)_a$. Then for every word $w \in A^\omega$ we have

$$\begin{aligned}(\|\forall x.\varphi\|, w) &= \prod_{i\geq 0}(\|\varphi\|, (w, [x \to i])) \\ &= \prod_{i\geq 0}\left(\left\|\bigvee_{a\in A}(k_a \wedge P_a(x))\right\|, (w, [x \to i])\right) \\ &= \prod_{i\geq 0}(r, w(i)) \\ &= (r^\omega, w)\end{aligned}$$

where the fourth equality holds by Example 2. Hence, we get $\|\forall x.\varphi\| = r^\omega$ which implies that $\|\forall x.\varphi\|$ is an ω-star-free series.

Next, let $\varphi = (y \leq x) \to \bigvee_{a\in A}(k_a \wedge P_a(x))$. We consider the subset $F = \{(a, 0) \mid a \in A\}$ of $A_{\{y\}}$. The language F^* is star-free, hence, the series 1_{F^*} is star-free. Consider the series $s = \sum_{a\in A}\left((k_a)_{(a,0)}\right)$ and $s' = \sum_{a\in A}\left((k_a)_{(a,1)}\right)$ over $A_{\{y\}}$ and K. Now for every $w \in A^\omega$ and $l \geq 0$, we get

$$\begin{aligned}(\|\forall x.\varphi\|, (w, [y \to l])) &= \prod_{j\geq 0}\left(\left\|(y \leq x) \to \bigvee_{a\in A}(k_a \wedge P_a(x))\right\|, (w, [x \to j, y \to l])\right) \\ &= \prod_{j\geq l}\left(\left\|\bigvee_{a\in A}(k_a \wedge P_a(x))\right\|, (w, [x \to j])\right) \\ &= (s', (w(l), 1)) \cdot \prod_{j>l}(s, (w(j), 0)) \\ &= (1_{F^*} \cdot (s' \cdot s^\omega), (w, [y \to l])),\end{aligned}$$

i.e., $\|\forall x.\varphi\| = 1_{F^*} \cdot (s' \cdot s^\omega)$ is an ω-star-free series.

Finally, let $\varphi = (y \leq x < z) \to \bigvee_{a\in A}(k_a \wedge P_a(x))$. We consider the finite languages $F = \{(a, 0, 0) \mid a \in A\}$, $F_1 = \{(a, 1, 0) \mid a \in A\}$, $F_2 = \{(a, 0, 1) \mid a \in A\}$ and $F_3 = \{(a, 1, 1) \mid a \in A\}$ over $A_{\{y,z\}}$. The languages $F, F_1, F_2, F_3, F^+, F^*$ are star-free, hence the series $1_{F_1}, 1_{F_2}, 1_{F_3}, 1_{F^+}, 1_{F^*}$ are star-free. Since $(F^+)^+ = F^+$ the languages $L = (F^+)^\omega, L' = F_2L$ are ω-star-free (cf. [28]) and the infinitary series $1_L, 1_{L'}$ are ω-star-free. We consider the series $s = \sum_{a\in A}\left(k_{(a,0,0)}\right)_{(a,0,0)}$ and $s' = \sum_{a\in A}\left(k_{(a,1,0)}\right)_{(a,1,0)}$ over $A_{\{y,z\}}$ and K, where $k_{(a,0,0)} = k_{(a,1,0)} = k_a$ for every $a \in A$. Moreover, we let $r_1 = 1_{F^*} \cdot (s' \cdot ((1_\varepsilon + s^+) \cdot 1_{L'}))$, $r_2 = 1_{F^*} \cdot (1_{F_3} \cdot 1_L)$, and $r_3 = 1_{F^*} \cdot (1_{F_2} \cdot (1_{F^*} \cdot (1_{F_1} \cdot 1_L)))$.

Now, for every $w \in A^\omega$ and $j, l \geq 0$ with $j < l$, we have

$(r_2 + r_3, (w, [y \to j, z \to l])) = 0$, and

$$\begin{aligned}
&(\|\forall x.\varphi\|, (w, [y \to j, z \to l])) \\
&= \prod_{i \geq 0} \left(\left\| (y \leq x < z) \to \bigvee\nolimits_{a \in A} (k_a \wedge P_a(x)) \right\|, (w, [x \to i, y \to j, z \to l]) \right) \\
&= \prod_{j \leq i < l} \left(\left\| \bigvee\nolimits_{a \in A} (k_a \wedge P_a(x)) \right\|, (w, [x \to i]) \right) \\
&= (s', (w(j), 1, 0)) \cdot \prod_{j < i < l} (s, (w(i), 0, 0)) \\
&= (r_1, (w, [y \to j, z \to l])) \\
&= (r_1 + (r_2 + r_3), (w, [y \to j, z \to l])) .
\end{aligned}$$

Furthermore, for every $w \in A^\omega$ and $j, l \geq 0$ with $j \geq l$, we get $(r_1, (w, [y \to j, z \to l])) = 0$, and

$$\begin{aligned}
&(\|\forall x.\varphi\|, (w, [y \to j, z \to l])) \\
&= \prod_{i \geq 0} \left(\left\| (y \leq x < z) \to \bigvee\nolimits_{a \in A} (k_a \wedge P_a(x)) \right\|, (w, [x \to i, y \to j, z \to l]) \right) \\
&= \prod_{i \geq 0} (\|\neg (y \leq x < z)\|, (w, [x \to i, y \to j, z \to l])) \\
&= (r_2 + r_3, (w, [y \to j, z \to l])) \\
&= (r_1 + (r_2 + r_3), (w, [y \to j, z \to l])) .
\end{aligned}$$

We conclude that $\|\forall x.\varphi\| = r_1 + (r_2 + r_3)$, hence $\|\forall x.\varphi\|$ is an ω-star-free series, as required. □

Now, we are ready to state the main result of the section.

Theorem 2. *ω-wqFO $(K, A) \subseteq \omega$-SF (K, A).*

Proof. We combine Lemmas 18, 19, 20, 21, and 22. □

8 Counter-free series

In this section, we consider the concept of counter-freeness within weighted (resp. weighted Büchi) automata over A and K. Our models will be nondeterministic. We need first to recall the notions of weighted automata and weighted Büchi automata over A and K. For simplicity reasons, we equip our finitary models with a set of final states instead of a terminal distribution.

A weighted automaton over A and K is a quadruple $\mathcal{A} = (Q, in, wt, F)$ where Q is the *finite state set*, $in : Q \to K$ is the *initial distribution*, $wt : Q \times A \times Q \to K$

is a mapping assigning *weights* to the transitions of the automaton and $F \subseteq Q$ is the *final state set*.

Given a word $w = a_0 \dots a_{n-1} \in A^*$, a path of $\mathcal{A}$ over w is a finite sequence of transitions $P_w := ((q_i, a_i, q_{i+1}))_{0 \leq i \leq n-1}$. The *running weight of* P_w is the value

$$rwt(P_w) := \prod_{0 \leq i \leq n-1} wt\left((q_i, a_i, q_{i+1})\right)$$

and the *weight of* P_w is given by

$$weight(P_w) := in(q_0) \cdot rwt(P_w).$$

The path P_w is called *successful* if $q_n \in F$. We denote by $\mathrm{succ}(\mathcal{A})$ the set of successful paths of $\mathcal{A}$. The *behavior of* $\mathcal{A}$ is the series $\|\mathcal{A}\| : A^* \to K$ which is defined, for every $w \in A^*$, by $(\|\mathcal{A}\|, w) = \sum_{P_w \in \mathrm{succ}(\mathcal{A})} weight(P_w)$. A series $r \in K\langle\langle A^* \rangle\rangle$ is called *recognizable* if it is the behavior of a weighted automaton over A and K.

A weighted Büchi automaton $\mathcal{A} = (Q, in, wt, F)$ over A and K is defined as a weighted automaton. Given an infinite word $w = a_0 a_1 \dots \in A^\omega$, a path of $\mathcal{A}$ over w is an infinite sequence of transitions $P_w := ((q_i, a_i, q_{i+1}))_{i \geq 0}$. The *running weight of* P_w is the value

$$rwt(P_w) := \prod_{i \geq 0} wt\left((q_i, a_i, q_{i+1})\right)$$

and the *weight of* P_w is given by

$$weight(P_w) := in(q_0) \cdot rwt(P_w).$$

A path P_w is called *successful* if at least one final state occurs infinitely often along P_w. Then, the *behavior of* $\mathcal{A}$ is the infinitary series $\|\mathcal{A}\| : A^\omega \to K$ whose coefficients are given by $(\|\mathcal{A}\|, w) = \sum_{P_w \in \mathrm{succ}(\mathcal{A})} weight(P_w)$, for every $w \in A^\omega$. An infinitary series $r \in K\langle\langle A^\omega \rangle\rangle$ is called *ω-recognizable* if it is the behavior of a weighted Büchi automaton over A and K.

We shall need also the following notation. Given a weighted (resp. weighted Büchi) automaton $\mathcal{A} = (Q, in, wt, F)$, a word $w = a_0 \dots a_{n-1} \in A^*$, and states $q, q' \in Q$, we shall denote by $P_{(q,w,q')}$ a path of $\mathcal{A}$ over w starting at state q and terminating at state q', i.e., $P_{(q,w,q')} = (q, a_0, q_1)\left((q_i, a_i, q_{i+1})\right)_{1 \leq i \leq n-2}(q_{n-1}, a_{n-1}, q')$. Then

$$rwt\left(P_{(q,w,q')}\right) = wt\left((q, a_0, q_1)\right) \cdot \prod\nolimits_{1 \leq i \leq n-2} wt\left((q_i, a_i, q_{i+1})\right) \cdot wt\left((q_{n-1}, a_{n-1}, q')\right).$$

Now, we are ready to introduce our counter-free weighted and counter-free weighted Büchi automata.

Definition 9. *A weighted automaton (resp. weighted Büchi automaton)* $\mathcal{A} = (Q, in, wt, F)$ *over* A *and* K *is called* counter-free *(*cfwa, *resp.* cfwBa*, for short) if for every* $q \in Q$*,* $w \in A^*$*, and* $n \geq 1$*, the relation* $\sum_{P_{(q,w^n,q)}} rwt\left(P_{(q,w^n,q)}\right) \neq 0$ *implies* $\sum_{P_{(q,w^n,q)}} rwt\left(P_{(q,w^n,q)}\right) = \left(\sum_{P_{(q,w,q)}} rwt\left(P_{(q,w,q)}\right)\right)^n$.

A series $r \in K\langle\langle A^*\rangle\rangle$ (resp. $r \in K\langle\langle A^\omega\rangle\rangle$) is called *counter-free* (resp. *ω-counter-free*) if it is accepted by a cfwa (resp. cfwBa) over A and K. We shall denote by $CF(K, A)$ (resp. ω-$CF(K, A)$) the class of all counter-free (resp. ω-counter-free) series over A and K.

A cfwa $\mathcal{A} = (Q, in, wt, F)$ over A and K is called *normalized* if there are two states $q_0, q_t \in Q$ such that $F = \{q_t\}$ and for every $q \in Q$, $a \in A$, we have $in(q) = 1$ if $q = q_0$, and $in(q) = 0$ otherwise, and $wt((q, a, q_0)) = 0 = wt((q_t, a, q))$. We denote a normalized cfwa $\mathcal{A}$ simply by $\mathcal{A} = (Q, q_0, wt, q_t)$.

The following result has been proved for weighted automata in [11].

Lemma 23. *For every cfwa* $\mathcal{A} = (Q, in, wt, F)$ *we can effectively construct a normalized cfwa* $\mathcal{A}' = (Q \cup \{q_0, q_t\}, q_0, wt', q_t)$ *such that* $(\|\mathcal{A}'\|, w) = (\|\mathcal{A}\|, w)$ *for every* $w \in A^+$ *and* $(\|\mathcal{A}'\|, \varepsilon) = 0$.

Proof. We use similar arguments as in the proof of Lm. 7 in [11]. In fact, it remains to show that the normalized weighted automaton $\mathcal{A}'$ is counter-free. Indeed, let $q \in Q \cup \{q_0, q_t\}$, $w \in A^+$, $n \geq 1$, and $P'_{(q,w^n,q)}$ be a path of $\mathcal{A}'$ over w with $rwt(P'_{(q,w^n,q)}) \neq 0$. Since $\mathcal{A}'$ is normalized we get that the states q_0, q_t do not occur in the path $P'_{(q,w^n,q)}$ hence $P'_{(q,w^n,q)}$ is also a path of $\mathcal{A}$. This implies that

$$\sum_{P'_{(q,w^n,q)}} rwt\left(P'_{(q,w^n,q)}\right) = \sum_{P_{(q,w^n,q)}} rwt\left(P_{(q,w^n,q)}\right) = \left(\sum_{P_{(q,w,q)}} rwt\left(P_{(q,w,q)}\right)\right)^n$$
$$= \left(\sum_{P'_{(q,w,q)}} rwt\left(P'_{(q,w,q)}\right)\right)^n,$$

where $P_{(q,w^n,q)}$ denotes a path of $\mathcal{A}$ over w, and this concludes our proof. □

A cfwBa $\mathcal{A} = (Q, in, wt, F)$ over A and K is called *initial weight normalized* if there is a state $q_0 \in Q$ such that for every $q \in Q$ and $a \in A$ we have $in(q) = 1$ if $q = q_0$, and $in(q) = 0$ otherwise, and $wt((q, a, q_0)) = 0$. We denote an initial weight normalized cfwBa $\mathcal{A}$ simply by $\mathcal{A} = (Q, q_0, wt, F)$.

Lemma 24. *For every cfwBa* $\mathcal{A} = (Q, in, wt, F)$ *we can effectively construct an initial weight normalized cfwBa* $\mathcal{A}' = (Q \cup \{q_0\}, q_0, wt', F)$ *such that* $\|\mathcal{A}'\| = \|\mathcal{A}\|$.

Proof. We use the same arguments, as in Lemma 23 for the modification of the initial distribution. □

In the sequel, we prove closure properties of the classes $CF(K,A)$ and ω-$CF(K,A)$. We shall need these properties in order to relate star-free and ω-star-free series with counter-free and ω-counter-free series, nevertheless, these results have also their own interest.

Proposition 6. *The class $CF(K,A)$ contains the monomials and it is closed under sum, Hadamard product, complement, Cauchy product, and iteration restricted to letter-step series.*

Proof. The closure of $CF(K,A)$ under sum, is shown by taking the disjoint union of two cfwa. In this case, any "loop" belongs either to the first or to the second automaton, hence the derived weighted automaton is also counter-free. Since monomials over A and K are obviously counter-free series, we get that letter-step series are also counter-free.

Closure under Hadamard product is proved by using the standard "product construction" of two cfwa. More precisely, let $\mathcal{A}_1=(Q_1, in_1, wt_1, F_1)$ and $\mathcal{A}_2=(Q_2, in_2, wt_2, F_2)$ be two cfwa over A and K. Consider the weighted automaton $\mathcal{A}=(Q, in, wt, F)$ with $Q = Q_1 \times Q_2$, $F = F_1 \times F_2$, and $in((q_1,q_2)) = in_1(q_1)\cdot in_2(q_2)$, $wt(((q_1,q_2),a,(p_1,p_2))) = wt_1((q_1,a,p_1))\cdot wt_2((q_2,a,p_2))$, for every $(q_1,q_2),(p_1,p_2)\in Q, a\in A$. Then, for every $w\in A^*$ and path P_w of $\mathcal{A}$ over w, there are two unique paths $P_{1,w}$ of $\mathcal{A}_1$ over w, and $P_{2,w}$ of $\mathcal{A}_2$ over w (obtained by projections of P_w on Q_1 and Q_2, respectively, in the obvious way) and vice-versa. Furthermore, we have $weight(P_w) = weight(P_{1,w})\cdot weight(P_{2,w})$. Now assume that for some $(q_1,q_2)\in Q$, $w\in A^*$, and $n\geq 1$ there is a path $P_{((q_1,q_2),w^n,(q_1,q_2))}$ with $rwt\left(P_{((q_1,q_2),w^n,(q_1,q_2))}\right)\neq 0$. Then

$$
\begin{aligned}
&\left(\sum_{P_{((q_1,q_2),w,(q_1,q_2))}} rwt\left(P_{((q_1,q_2),w,(q_1,q_2))}\right)\right)^n\\
&=\left(\sum_{P_{1,(q_1,w,q_1)},P_{2,(q_2,w,q_2)}} \left(rwt\left(P_{1,(q_1,w,q_1)}\right)\cdot rwt\left(P_{2,(q_2,w,q_2)}\right)\right)\right)^n\\
&=\left(\sum_{P_{1,(q_1,w,q_1)}} rwt\left(P_{1,(q_1,w,q_1)}\right)\cdot \sum_{P_{2,(q_2,w,q_2)}} rwt\left(P_{2,(q_2,w,q_2)}\right)\right)^n\\
&=\left(\sum_{P_{1,(q_1,w,q_1)}} rwt\left(P_{1,(q_1,w,q_1)}\right)\right)^n\cdot\left(\sum_{P_{2,(q_2,w,q_2)}} rwt\left(P_{2,(q_2,w,q_2)}\right)\right)^n\\
&=\sum_{P_{1,(q_1,w^n,q_1)}} rwt\left(P_{1,(q_1,w^n,q_1)}\right)\cdot \sum_{P_{2,(q_2,w^n,q_2)}} rwt\left(P_{2,(q_2,w^n,q_2)}\right)\\
&=\sum_{P_{((q_1,q_2),w^n,(q_1,q_2))}} rwt\left(P_{((q_1,q_2),w^n,(q_1,q_2))}\right)
\end{aligned}
$$

which implies that $\mathcal{A}$ is counter-free, and by construction $\|\mathcal{A}\| = \|\mathcal{A}_1\|\odot\|\mathcal{A}_2\|$.

Next, let $r \in CF(K,A)$ and $\mathcal{A} = (Q, in, wt, F)$ be a cfwa accepting r. We consider the nondeterministic finite automaton $\mathcal{A}' = (Q, A, I, \Delta, F)$ with $I = \{q \in Q \mid in(q) \neq 0\}$ and $\Delta = \{(q,a,q') \in Q \times A \times Q \mid wt((q,a,q')) \neq 0\}$. By construction of $\mathcal{A}'$, and since K is zero-divisor free, we get that for every $q_1, q_2 \in Q$ and $w \in A^*$ the path $P_{(q_1,w,q_2)}$ exists in $\mathcal{A}'$ iff $rwt(P_{(q_1,w,q_2)}) \neq 0$ in $\mathcal{A}$. Therefore, $\mathcal{A}'$ accepts the language $\mathrm{supp}(r)$ and it is trivially counter-free hence, $\mathrm{supp}(r)$ is a counter-free language. Then, $\overline{\mathrm{supp}(r)}$ is a counter-free language and let $\mathcal{B}$ be a counter-free automaton accepting it. We convert $\mathcal{B}$, in the obvious way, to a weighted automaton $\mathcal{B}'$ (with weights only 0 and 1) over A and K. Since K is idempotent, $\mathcal{B}'$ trivially accepts $1_{\overline{\mathrm{supp}(r)}} = \overline{r}$, and it is easily obtained that it is counter-free. We conclude that the series $\overline{r}$ is counter-free, as required.

Let now $\mathcal{A}_1{=}(Q_1, in_1, wt_1, F_1)$ and $\mathcal{A}_2{=}(Q_2, in_2, wt_2, F_2)$ be two cfwa over A and K. Using Lemma 23 we consider the normalized cfwa $\mathcal{A}_1'{=}(Q_1 \cup \{q_{0,1}, q_{t,1}\}, q_{0,1}, wt_1', q_{t,1})$ and $\mathcal{A}_2'{=}(Q_2 \cup \{q_{0,2}, q_{t,2}\}, q_{0,2}, wt_2', q_{t,2})$ such that $\|\mathcal{A}_i'\|$ coincides with $\|\mathcal{A}_i\|$ on A^+ for $i = 1, 2$. Without any loss, we assume that $(Q_1 \cup \{q_{0,1}, q_{t,1}\}) \cap (Q_2 \cup \{q_{0,2}, q_{t,2}\}) = \emptyset$. We construct the weighted automaton $\mathcal{A} {=}(Q, q_{0,1}, wt, q_{t,2})$ with $Q = Q_1 \cup \{q_{0,1}\} \cup Q_2 \cup \{q_{0,2}, q_{t,2}\}$ where we identify the states $q_{t,1}$ and $q_{0,2}$, and define the weight assignment mapping wt for every $q, q' \in Q, a \in A$ by

$$wt((q,a,q')) = \begin{cases} wt_1'((q,a,q')) & \text{if } q,q' \in Q_1 \cup \{q_{0,1}\} \\ wt_2'((q,a,q')) & \text{if } q,q' \in Q_2 \cup \{q_{0,2}, q_{t,2}\} \\ wt_1'((q,a,q_{t,1})) & \text{if } q \in Q_1 \cup \{q_{0,1}\} \text{ and } q' = q_{0,2} \\ 0 & \text{otherwise.} \end{cases}$$

It is a routine matter to formally prove that $\|\mathcal{A}\| = \|\mathcal{A}_1'\| \cdot \|\mathcal{A}_2'\|$. Furthermore, the weighted automaton $\mathcal{A}$ is counter-free since, by construction, any "loop" with weight $\neq 0$ belongs either to $\mathcal{A}_1'$ or to $\mathcal{A}_2'$. Now we let $k_i = (\|\mathcal{A}_i\|, \varepsilon)$ for $i = 1, 2$. Then $\|\mathcal{A}_1\| \cdot \|\mathcal{A}_2\| = (\|\mathcal{A}_1'\| \cdot \|\mathcal{A}_2'\|) + ((k_1)_\varepsilon \cdot \|\mathcal{A}_2'\|) + (\|\mathcal{A}_1'\| \cdot (k_2)_\varepsilon) + ((k_1)_\varepsilon \cdot (k_2)_\varepsilon)$. One can trivially construct cfwa accepting $(k_1)_\varepsilon$ and $(k_2)_\varepsilon$ and using simplifications of our previous construction[2] for $\mathcal{A}$ can easily show that the series $(k_1)_\varepsilon \cdot \|\mathcal{A}_2'\|$, $\|\mathcal{A}_1'\| \cdot (k_2)_\varepsilon$, and $(k_1)_\varepsilon \cdot (k_2)_\varepsilon$ are counter-free which implies, by what we have shown, that $\|\mathcal{A}_1\| \cdot \|\mathcal{A}_2\|$ is a counter-free series.

Finally, let $r = \sum_{a \in A} (k_a)_a$ be a letter-step series with $k_a \in K$ for every $a \in A$. We consider the cfwa $\mathcal{A} = (\{q_0, q_t\}, q_0, wt, q_t)$ with $wt((q_0,a,q_t)) = wt((q_t,a,q_t)) = k_a$ for every $a \in A$, and the weight of any other transition is 0. Obviously $r^+ = \|\mathcal{A}\|$, and we are done. □

Proposition 7. *The class ω-$CF(K,A)$ is closed under sum, complement, Cauchy product and ω-iteration restricted to letter-step series.*

Proof. The closure under sum and complement is shown as in Proposition 6. In particular, for the complement we use the property $k \neq 0 \implies \prod_{i \geq 0} k \neq 0$ for every $k \in K$, the fact that the class of counter-free Büchi recognizable (i.e., ω-star-free) languages is closed under complement (cf. [7]), and Lemma 1(i).

[2] In fact the cfwa for $(k_1)_\varepsilon$ and $(k_2)_\varepsilon$ cannot be normalized.

Next, let $s_1 \in CF(K,A)$ and $s_2 \in \omega\text{-}CF(K,A)$, and $\mathcal{A}_1 = (Q_1, in_1, wt_1, F_1)$, $\mathcal{A}_2 = (Q_2, in_2, wt_2, F_2)$ be a cfwa and a cfwBa over A and K accepting s_1 and s_2, respectively. Furthermore, let $\mathcal{A}'_1{=}(Q_1 \cup \{q_{0,1}, q_t\}, q_{0,1}, wt'_1, q_t)$ be the normalized automaton derived by $\mathcal{A}_1$ (cf. Lemma 23), and $\mathcal{A}'_2{=}(Q_2 \cup \{q_{0,2}\}, q_{0,2}, wt'_2, F_2)$ be the initial weight normalized cfwBa derived by $\mathcal{A}_2$ (cf. Lemma 24). Without any loss, we assume that $(Q_1 \cup \{q_{0,1}, q_t\}) \cap (Q_2 \cup \{q_{0,2}\}) = \emptyset$. Consider the weighted automaton $\mathcal{A} = (Q, q_{0,1}, wt, F_2)$ with $Q = Q_1 \cup \{q_{0,1}\} \cup Q_2 \cup \{q_{0,2}\}$ where we have identified the states q_t and $q_{0,2}$. The weight assignment mapping wt is defined for every $q, q' \in Q$ and $a \in A$ by

$$wt((q,a,q')) = \begin{cases} wt'_1((q,a,q')) & \text{if } q,q' \in Q_1 \cup \{q_{0,1}\} \\ wt'_2((q,a,q')) & \text{if } q,q' \in Q_2 \cup \{q_{0,2}\} \\ wt'_1((q,a,q_t)) & \text{if } q \in Q_1 \cup \{q_{0,1}\} \text{ and } q' = q_{0,2} \\ 0 & \text{otherwise.} \end{cases}$$

Trivially, $\|\mathcal{A}\| = s_1|_{A^+} \cdot s_2$. Furthermore, the weighted Büchi automaton $\mathcal{A}$ is counter-free since every "loop" with weight $\neq 0$ belongs either to $\mathcal{A}'_1$ or to $\mathcal{A}'_2$. Let $(s_1, \varepsilon) = k$. Then $s_1 \cdot s_2 = s_1|_{A^+} \cdot s_2 + k_\varepsilon \cdot s_2$ which concludes our claim since $k_\varepsilon \cdot s_2$ is trivially ω-counter-free.

Finally, let $r = \sum_{a \in A} (k_a)_a$ be a letter-step series with $k_a \in K$ for every $a \in A$. We consider the initial weight normalized cfwBa $\mathcal{A} = (\{q_0, q_t\}, q_0, wt, \{q_t\})$ with $wt((q_0,a,q_t)) = wt((q_t,a,q_t)) = k_a$ for every $a \in A$, and the weight of any other transition is 0. Obviously $r^\omega = \|\mathcal{A}\|$, and our proof is completed. □

Next, we introduce the subclass of almost simple counter-free (resp. almost simple ω-counter-free) series and we show, in Section 9, that it contains the class $SF(K,A)$ (resp. $\omega\text{-}SF(K,A)$).

Definition 10. *A cfwa (resp. cfwBa) $\mathcal{A} = (Q, in, wt, F)$ over A and K is called* simple *if for every $q, q', p, p' \in Q$, and $a \in A$, $in(q) \neq 0 \neq in(q')$ implies $in(q) = in(q')$, and $wt((q,a,q')) \neq 0 \neq wt((p,a,p'))$ implies $wt((q,a,q')) = wt((p,a,p'))$. Furthermore, a series $r \in K\langle\langle A^* \rangle\rangle$ (resp. $r \in K\langle\langle A^\omega \rangle\rangle$) is* simple *if it is the behavior of a simple cfwa (resp. cfwBa) over A and K.*

Proposition 8. *If $r, s \in K\langle\langle A^\omega \rangle\rangle$ are simple infinitary series, then $r \odot s$ is also simple.*

Proof. Let $\mathcal{A}, \mathcal{B}$ be two simple cfwBa accepting r, s, respectively. We let k, l for the weights $\neq 0$ assigned by the initial distributions of $\mathcal{A}, \mathcal{B}$, respectively, and k_a, l_a for the weights $\neq 0$ of the transitions labelled by $a \in A$, in $\mathcal{A}$ and $\mathcal{B}$, respectively. Without any loss, we assume that k_a, l_a exist for every $a \in A$, otherwise we consider a subalphabet of A. The language $L = \mathrm{supp}(\|\mathcal{A}\|) \cap \mathrm{supp}(\|\mathcal{B}\|)$ is ω-counter-free (cf. the proof of Proposition 7), and we get

$$\|\mathcal{A}\| \odot \|\mathcal{B}\| = 1_L \odot \left((k \cdot l) \left(\sum_{a \in A} (k_a \cdot l_a)_a \right)^\omega \right).$$

Let $\mathcal{C} = (Q, A, I, \Delta, F)$ be a counter-free nondeterministic Büchi automaton accepting L and consider the wBa $\mathcal{C}'=(Q, in, wt, F)$ where for every $q, q' \in Q, a \in A$ we let $in\,(q) = k \cdot l$ if $q \in I$, and $in\,(q) = 0$ otherwise, and $wt\,((q, a, q')) = k_a \cdot l_a$ if $(q, a, q') \in \Delta$, and $wt\,((q, a, q')) = 0$ otherwise. Since $\mathcal{C}$ is counter-free, we can easily show, using the idempotency property of K, that $\mathcal{C}'$ is also counter-free. Moreover, by definition $\mathcal{C}'$ is simple, and $\|\mathcal{C}'\| = \|\mathcal{A}\| \odot \|\mathcal{B}\|$ which concludes our proof. □

Definition 11.

- *A series $r \in K\langle\langle A^*\rangle\rangle$ is called* almost simple *if $r = \sum_{1\leq i\leq n} \left(r_1^{(i)} \cdot \ldots \cdot r_{m_i}^{(i)}\right)$ where, for every $1 \leq i \leq n$, $r_1^{(i)}, \ldots, r_{m_i}^{(i)}$ are simple counter-free series over A and K.*
- *A series $r \in K\langle\langle A^\omega\rangle\rangle$ is called* almost simple *if $r = \sum_{1\leq i\leq n} \left(r_1^{(i)} \cdot \ldots \cdot r_{m_i}^{(i)}\right)$ where, for every $1 \leq i \leq n$, $r_1^{(i)}, \ldots, r_{m_i-1}^{(i)}$ are simple counter-free series and $r_{m_i}^{(i)}$ is a simple ω-counter-free series over A and K.*

From the above definition and Proposition 6 (resp. Proposition 7), we get that a finitary (resp. infinitary) almost simple series is a counter-free (resp. an ω-counter-free) series[3]. We shall denote by $asCF(K, A)$ the class of almost simple counter-free series and by ω-$asCF(K, A)$ the class of almost simple ω-counter-free series over A and K.

9 ω-star-free series are almost simple ω-counter-free

In this section we prove that every star-free (resp. ω-star-free) series is an almost simple counter-free (resp. almost simple ω-counter-free) series.

Theorem 3. $SF(K, A) \subseteq asCF(K, A)$.

Proof. The class $asCF(K, A)$ trivially contains the monomials over A and K. Therefore, it suffices to show that it is closed under sum, Hadamard product, complement, Cauchy product, and iteration restricted to letter-step series.

Closure under sum and Cauchy product is easily obtained by definition of the class of almost simple counter-free series. For the closure under complement, let $r \in asCF(K, A)$, i.e., $r \in CF(K, A)$. Then the weighted automaton $\mathcal{B}'$ in the proof of Proposition 6 is simple and moreover accepts the complement $\overline{r}$ hence, $\overline{r} \in asCF(K, A)$. Trivially, we get that $asCF(K, A)$ contains the letter-step series. Furthermore, the automaton $\mathcal{A}$ accepting r^+ for a letter-step series r, in the proof of Proposition 6, is trivially simple, hence the class $asCF(K, A)$ is closed under iteration restricted to letter-step series. Therefore, it remains to prove

[3] In fact we can define an *almost simple* counter-free weighted (resp. weighted Büchi) automaton, but we do not need it here.

the closure under $\odot$. Since, $\odot$ distributes over sum it suffices to show that if $\mathcal{A}_i = (Q_i, in_i, wt_i, F_i), \mathcal{B}_j = (P_j, in'_j, wt'_j, T_j)$, for $1 \leq i \leq n, 1 \leq j \leq m$, are simple cfwa over A and K, then the counter-free series $(\|\mathcal{A}_1\| \cdot \ldots \cdot \|\mathcal{A}_n\|) \odot (\|\mathcal{B}_1\| \cdot \ldots \cdot \|\mathcal{B}_m\|)$ is almost simple. We proceed by induction on m, hence, assume firstly that $m = 1$. Without any loss, we suppose the state sets Q_i $(1 \leq i \leq n)$ to be pairwise disjoint[4]. For every $p, p' \in P_1$ and $2 \leq i \leq n-1$, we consider the simple cfwa $\mathcal{C}_{1,p} = (Q_1 \times P_1, \overline{in}_1, \overline{wt}_1, F_1 \times \{p\})$, $\mathcal{C}_{i,(p,p')} = (Q_i \times P_1, \overline{in}_{i,(p,p')}, \overline{wt}_i, F_i \times \{p'\})$, and $\mathcal{C}_{n,p} = (Q_n \times P_1, \overline{in}_{n,p}, \overline{wt}_n, F_n \times T_1)$ by

- $\overline{in}_1\left(\left(q^{(1)}, p_1\right)\right) = in_1(q^{(1)}) \cdot in'_1(p_1)$ for every $q^{(1)} \in Q_1, p_1 \in P_1$,
- $\overline{wt}_1\left(\left((q_1^{(1)}, p_1), a, (q_2^{(1)}, p_2)\right)\right) = wt_1\left(\left(q_1^{(1)}, a, q_2^{(1)}\right)\right) \cdot wt'_1\left((p_1, a, p_2)\right)$ for every $q_1^{(1)}, q_2^{(1)} \in Q_1, p_1, p_2 \in P_1, a \in A$, and

for every $2 \leq i \leq n-1$

- $\overline{in}_{i,(p,p')}\left(\left(q^{(i)}, p_1\right)\right) = in_i(q^{(i)})$ if $p_1 = p$, and $\overline{in}_{i,(p,p')}\left(\left(q^{(i)}, p_1\right)\right) = 0$ otherwise, for every $q^{(i)} \in Q_i, p_1 \in P_1$,
- $\overline{wt}_i\left(\left((q_1^{(i)}, p_1), a, (q_2^{(i)}, p_2)\right)\right) = wt_i\left(\left(q_1^{(i)}, a, q_2^{(i)}\right)\right) \cdot wt'_1((p_1, a, p_2))$ for every $q_1^{(i)}, q_2^{(i)} \in Q_i, p_1, p_2 \in P_1, a \in A$, and
- $\overline{in}_{n,p}\left(\left(q^{(n)}, p_1\right)\right) = in_n(q^{(n)})$ if $p_1 = p$, and $\overline{in}_{n,p}\left(\left(q^{(n)}, p_1\right)\right) = 0$ otherwise, for every $q^{(n)} \in Q_n, p_1 \in P_1$,
- $\overline{wt}_n\left(\left((q_1^{(n)}, p_1), a, (q_2^{(n)}, p_2)\right)\right) = wt_n\left(\left(q_1^{(n)}, a, q_2^{(n)}\right)\right) \cdot wt'_1((p_1, a, p_2))$, for every $q_1^{(n)}, q_2^{(n)} \in Q_n, p_1, p_2 \in P_1, a \in A$.

We claim that

$$(\|\mathcal{A}_1\| \cdot \ldots \cdot \|\mathcal{A}_n\|) \odot \|\mathcal{B}_1\| = \sum_{p_1, \ldots, p_{n-1} \in P_1} \left(\|\mathcal{C}_{1,p_1}\| \cdot \left\|\mathcal{C}_{2,(p_1,p_2)}\right\| \cdot \ldots \cdot \left\|\mathcal{C}_{n-1,(p_{n-2},p_{n-1})}\right\| \cdot \left\|\mathcal{C}_{n,p_{n-1}}\right\|\right).$$

Clearly, it suffices to prove that for every $w \in A^*$, the sum

$$\left(\sum_{w=w_1 \ldots w_n} \left(\prod_{1 \leq i \leq n} \left(\sum_{P_{w_i}^{(i)} \in \mathrm{succ}(\mathcal{A}_i)} weight\left(P_{w_i}^{(i)}\right)\right)\right)\right)\left(\sum_{P_w \in \mathrm{succ}(\mathcal{B}_1)} weight\left(P_w\right)\right)$$

[4]Here, we deal with the case $n > 1$. For $n = m = 1$ we consider the product automaton of two simple cfwa which is trivially simple.

equals to

$$\sum_{p_1,\dots,p_{n-1}\in P_1}\left(\sum_{w=w_1\dots w_n}\left(\begin{array}{c}\displaystyle\sum_{\overline{P}_{w_1}\in \mathrm{succ}(\mathcal{C}_{1,p_1})} weight(\overline{P}_{w_1})\\ \left(\displaystyle\prod_{1\le i\le n-2}\left(\sum_{\overline{P}_{w_i}\in \mathrm{succ}\left(\mathcal{C}_{i,(p_i,p_{i+1})}\right)} weight(\overline{P}_{w_i})\right)\right)\\ \displaystyle\sum_{\overline{P}_{w_{n-1}}\in \mathrm{succ}(\mathcal{C}_{n,p_{n-1}})} weight(\overline{P}_{w_{n-1}})\end{array}\right)\right).$$

To this end, let $w = a_0a_1\dots a_{m-1} \in \mathrm{supp}((\|\mathcal{A}_1\|\cdot\ldots\cdot\|\mathcal{A}_n\|)\odot\|\mathcal{B}_1\|)$ with $a_0,a_1,\dots,a_{m-1}\in A$. Let us assume that $w_1,\dots,w_n\in A^*$, with $w = w_1\dots w_n$, and $P^{(1)}_{w_1} : \left(q^{(1)}_0,a_0,q^{(1)}_1\right)\left(q^{(1)}_1,a_1,q^{(1)}_2\right)\dots\left(q^{(1)}_{i_1},a_{i_1},q^{(1)}_{i_1+1}\right)$,

$P^{(2)}_{w_2} : \left(q^{(2)}_{i_1+1},a_{i_1+1},q^{(2)}_{i_1+2}\right)\left(q^{(2)}_{i_1+2},a_{i_1+2},q^{(2)}_{i_1+3}\right)\dots\left(q^{(2)}_{i_2},a_{i_2},q^{(2)}_{i_2+1}\right)$,

$\vdots$

$P^{(n)}_{w_n} : \left(q^{(n)}_{i_{n-1}+1},a_{i_{n-1}+1},q^{(n)}_{i_{n-1}+2}\right)\left(q^{(n)}_{i_{n-1}+2},a_{i_{n-1}+2},q^{(n)}_{i_{n-1}+3}\right)\dots\left(q^{(n)}_{m-1},a_{m-1},q^{(n)}_{m}\right)$,

and $P_w : (p_0,a_0,p_1)(p_1,a_1,p_2)\dots(p_{m-1},a_{m-1},p_m)$,

are successful paths of $\mathcal{A}_1,\mathcal{A}_2\dots,\mathcal{A}_n,\mathcal{B}_1$ over $w_1,\dots,w_n,w$ respectively. By definition of $\mathcal{C}_{1,p_{i_1+1}},\mathcal{C}_{2,(p_{i_1+1},p_{i_2+1})},\dots,\mathcal{C}_{n,p_{i_{n-1}+1}}$, we can construct from $P^{(1)}_{w_1},\dots,P^{(n)}_{w_n}$ and P_w the paths $\overline{P}_{w_1},\dots,\overline{P}_{w_n}$ of $\mathcal{C}_{1,p_{i_1+1}},\dots,\mathcal{C}_{n,p_{i_{n-1}+1}}$ over $w_1,\dots,w_n$ respectively, as follows.

$\overline{P}_{w_1} : \left(\left(q^{(1)}_0,p_0\right),a_0,\left(q^{(1)}_1,p_1\right)\right)\left(\left(q^{(1)}_1,p_1\right),a_1,\left(q^{(1)}_2,p_2\right)\right)\dots\left(\left(q^{(1)}_{i_1},p_{i_1}\right),a_{i_1},\left(q^{(1)}_{i_1+1},p_{i_1+1}\right)\right)$,

$\overline{P}_{w_2} : \left(\left(q^{(2)}_{i_1+1},p_{i_1+1}\right),a_{i_1+1},\left(q^{(2)}_{i_1+2},p_{i_1+2}\right)\right)\left(\left(q^{(2)}_{i_1+2},p_{i_1+2}\right),a_{i_1+2},\left(q^{(2)}_{i_1+3},p_{i_1+3}\right)\right)\dots\left(\left(q^{(2)}_{i_2},p_{i_2}\right),a_{i_2},\left(q^{(2)}_{i_2+1},p_{i_2+1}\right)\right)$,

$\vdots$

$\overline{P}_{w_n} : \left(\left(q^{(n)}_{i_{n-1}+1},p_{i_{n-1}+1}\right),a_{i_{n-1}+1},\left(q^{(n)}_{i_{n-1}+2},p_{i_{n-1}+2}\right)\right)\left(\left(q^{(n)}_{i_{n-1}+2},p_{i_{n-1}+2}\right),a_{i_{n-1}+2},\left(q^{(n)}_{i_{n-1}+3},p_{i_{n-1}+3}\right)\right)\dots\left(\left(q^{(n)}_{m-1},p_{m-1}\right),a_{m-1},\left(q^{(n)}_{m},p_m\right)\right)$.

Then, $weight\left(\overline{P}_{w_1}\right)\cdot weight\left(\overline{P}_{w_2}\right)\cdot\ldots\cdot weight\left(\overline{P}_{w_n}\right) = weight\left(P^{(1)}_{w_1}\right)\cdot weight\left(P^{(2)}_{w_2}\right)\cdot\ldots\cdot weight\left(P^{(n)}_{w_n}\right)\cdot weight(P_w)$. Conversely, let $p_{i_1+1},\dots,p_{i_{n-1}+1}\in P_1$ such that $w\in\mathrm{supp}\left(\left\|\mathcal{C}_{1,p_{i_1+1}}\right\|\cdot\ldots\cdot\left\|\mathcal{C}_{n,p_{i_{n-1}+1}}\right\|\right)$. Using similar arguments as above, and keeping the previous notations, we get that for every $w_1,\dots,w_n\in A^*$ with $w = w_1\dots w_n$, and successful paths $\overline{P}_{w_1},\overline{P}_{w_2},\dots,\overline{P}_{w_n}$, there exist successful paths $P^{(1)}_{w_1},P^{(2)}_{w_2},\dots,P^{(n)}_{w_n},P_w$ such that $weight\left(\overline{P}_{w_1}\right)\cdot weight\left(\overline{P}_{w_2}\right)\cdot\ldots\cdot weight\left(\overline{P}_{w_n}\right) = weight\left(P^{(1)}_{w_1}\right)\cdot weight\left(P^{(2)}_{w_2}\right)\cdot\ldots\cdot weight\left(P^{(n)}_{w_n}\right)\cdot weight(P_w)$. Therefore, by standard computations, we get the equality of the two sums and this concludes our claim for $m = 1$.

For the induction step, for simplicity, we prove our claim for $m = 2$. For every $1\le i\le n$ and $q^{(i)}\in Q_i$, we define the simple cfwa $\mathcal{A}_{i,q^{(i)}} = (Q_i,in_i,wt_i,\{q^{(i)}\})$

and $\mathcal{A}'_{i,q^{(i)}} = (Q_i, in'_i, wt_i, F_i)$ with $in'_i(q) = 1$ if $q = q^{(i)}$, and $in'_i(q) = 0$ otherwise, for every $q \in Q_i$. Then, with similar as above arguments, we can show that $(\|\mathcal{A}_1\| \cdot \ldots \cdot \|\mathcal{A}_n\|) \odot (\|\mathcal{B}_1\| \cdot \|\mathcal{B}_2\|)$ equals to

$$\sum_{\substack{1 \leq i \leq n \\ q^{(i)} \in Q_i}} \left(\left(\left(\|\mathcal{A}_1\| \cdot \ldots \cdot \left\|\mathcal{A}_{i,q^{(i)}}\right\| \right) \odot \|\mathcal{B}_1\| \right) \cdot \left(\left(\left\|\mathcal{A}'_{i,q^{(i)}}\right\| \cdot \ldots \cdot \|\mathcal{A}_n\| \right) \odot \|\mathcal{B}_2\| \right) \right).$$

Hence, by induction hypothesis we conclude our claim. □

Below, in our second main result of the present section, we show that every ω-star-free series is an almost simple ω-counter-free series.

Theorem 4. *ω-$SF(K, A) \subseteq \omega$-$asCF(K, A)$.*

Proof. By Definition 8 and Theorem 3, it suffices to show that the class ω-$asCF(A, K)$ is closed under sum, Hadamard product, complement, ω-iteration restricted to letter-step series, and if $s_1 \in asCF(K, A)$ and $s_2 \in \omega$-$asCF(K, A)$, then $s_1 \cdot s_2 \in \omega$-$asCF(K, A)$. The last property as well as closure under sum are easily obtained by Definition 11. For the closure under complement, we use a similar argument as in the corresponding part of the proof of Theorem 3. Furthermore, the automaton $\mathcal{A}$ accepting r^ω for a letter-step series r, in the proof of Proposition 7, is trivially simple, hence the class ω-$asCF(K, A)$ is closed under ω-iteration restricted to letter-step series. Again, the most complicated case is to prove the closure under Hadamard product, i.e., to prove that if $\mathcal{A}_i = (Q_i, in_i, wt_i, F_i), \mathcal{B}_j = (P_j, in'_j, wt'_j, T_j)$, for $1 \leq i \leq n-1, 1 \leq j \leq m-1$, are simple cfwa and $\mathcal{A}_n = (Q_n, in_n, wt_n, F_n), \mathcal{B}_m = (P_m, in'_m, wt'_m, T_m)$ are simple cfwBa over A and K, then the ω-counter-free series $(\|\mathcal{A}_1\| \cdot \ldots \cdot \|\mathcal{A}_n\|) \odot (\|\mathcal{B}_1\| \cdot \ldots \cdot \|\mathcal{B}_m\|)$ is almost simple. We state our proof by induction on m, hence, let firstly $m = 1$, i.e., $\mathcal{B}_1 = (P_1, in'_1, wt'_1, T_1)$ be a simple cfwBa (again we assume $n > 1$, otherwise if $n = m = 1$ we get our result by Proposition 8). We keep the notations of Theorem 3 and consider the simple cfwa $\mathcal{C}_{1,p}$, and $\mathcal{C}_{i,(p,p')}$ for every $2 \leq i \leq n-1$. Furthremore, for every $p \in P_1$ we define the wBa $\mathcal{C}_{n,p} = \left(Q_n \times P_1 \times \{0,1,2\}, \overline{in}_{n,p}, \overline{wt}_n, Q_n \times P_1 \times \{2\}\right)$ with the initial distribution $\overline{in}_{n,p}$ given for every $q^{(n)} \in Q_n, p_1 \in P_1, x \in \{0,1,2\}$ by

$$\overline{in}_{n,p}(q^{(n)}, p_1, x) = \begin{cases} in_n(q^{(n)}) & \text{if } p_1 = p, x = 0 \\ 0 & \text{otherwise} \end{cases},$$

and the weight assignment mapping $\overline{wt}_n$ defined for every $q_1^{(n)}, q_2^{(n)} \in Q_n, p_1, p_2 \in P_1, a \in A, x, y \in \{0,1,2\}$ as follows.

$$\overline{wt}_n\left(\left(\left(q_1^{(n)}, p_1, x\right), a, \left(q_2^{(n)}, p_2, y\right)\right)\right) = wt_n\left(\left(q_1^{(n)}, a, q_2^{(n)}\right)\right) \cdot wt'_1\left((p_1, a, p_2)\right)$$

if $(x = y = 0$ or $q_2^{(n)} \in F_n, x = 0, y = 1$ or $p_2 \notin T_1, x = y = 1$ or $p_2 \in T_1, x = 1, y = 2$ or $x = 2, y = 0)$, and

$\overline{wt}_n\left(\left(\left(q_1^{(n)}, p_1, x\right), a, \left(q_2^{(n)}, p_2, y\right)\right)\right) = 0$ otherwise[5].

We note that, since $\mathcal{A}_n$ (resp. $\mathcal{B}_1, \mathcal{C}_{n,p}$)[6] is simple, for every $w \in A^\omega$, all the successful paths of $\mathcal{A}_n$ (resp. $\mathcal{B}_1, \mathcal{C}_{n,p}$) over w with weight $\neq 0$ have the same weight. Again we will show that

$$(\|\mathcal{A}_1\| \cdot \ldots \cdot \|\mathcal{A}_n\|) \odot \|\mathcal{B}_1\| = \sum_{p_1,\ldots,p_{n-1} \in P_1} \left(\|\mathcal{C}_{1,p_1}\| \cdot \left\|\mathcal{C}_{2,(p_1,p_2)}\right\| \cdot \ldots \cdot \left\|\mathcal{C}_{n-1,(p_{n-2},p_{n-1})}\right\| \cdot \left\|\mathcal{C}_{n,p_{n-1}}\right\|\right)$$

by proving that for every $w \in A^\omega$ the sum

$$\left(\sum_{\substack{w=w_1\ldots w_n \\ w_1,\ldots,w_{n-1} \in A^*, w_n A^\omega}} \left(\prod_{1\le i\le n} \left(\sum_{P_{w_i}^{(i)} \in \mathrm{succ}(\mathcal{A}_i)} weight\left(P_{w_i}^{(i)}\right)\right)\right)\right)\left(\sum_{P_w \in \mathrm{succ}(\mathcal{B}_1)} weight\left(P_w\right)\right)$$

equals to

$$\sum_{p_1,\ldots,p_{n-1} \in P_1} \left(\sum_{\substack{w=w_1\ldots w_n \\ w_1,\ldots,w_{n-1} \in A^*, w_n A^\omega}} \left(\begin{array}{c} \sum\limits_{\overline{P}_{w_1} \in \mathrm{succ}\left(\mathcal{C}_{1,p_1}\right)} weight(\overline{P}_{w_1}) \\ \left(\prod\limits_{1\le i\le n-2} \left(\sum\limits_{\overline{P}_{w_i} \in \mathrm{succ}\left(\mathcal{C}_{i,(p_i,p_{i+1})}\right)} weight(\overline{P}_{w_i})\right)\right) \\ \sum\limits_{\overline{P}_{w_{n-1}} \in \mathrm{succ}\left(\mathcal{C}_{n,p_{n-1}}\right)} weight(\overline{P}_{w_{n-1}}) \end{array}\right)\right).$$

To this end, let $w = a_0 a_1 \ldots \in \mathrm{supp}((\|\mathcal{A}_1\| \cdot \ldots \cdot \|\mathcal{A}_n\|) \odot \|\mathcal{B}_1\|)$ with $a_0, a_1, \ldots \in A$. We fix an analysis $w = w_1 \ldots w_{n-1} w_n$ $(w_1, \ldots, w_{n-1} \in A^*, w_n \in A^\omega)$, and we let $P_{w_i}^{(i)}$, for every $1 \le i \le n$, to be a successful path of $\mathcal{A}_i$ over w_i, and P_w a successful path of $\mathcal{B}_1$ over w. We keep the notations of the proof of Theorem 3, for the paths $P_{w_i}^{(i)}$ $(1 \le i \le n-1)$, and we set

$P_{w_n}^{(n)} : \left(q_{i_{n-1}+1}^{(n)}, a_{i_{n-1}+1}, q_{i_{n-1}+2}^{(n)}\right)\left(q_{i_{n-1}+2}^{(n)}, a_{i_{n-1}+2}, q_{i_{n-1}+3}^{(n)}\right)\ldots,$ and

$P_w : (p_0, a_0, p_1)(p_1, a_1, p_2)\ldots.$

We consider the paths $\overline{P}_{w_i}$ $(1 \le i \le n-1)$ as in the proof of Theorem 3, and let

$$\overline{P}_{w_n} : \left(\left(q_{i_{n-1}+1}^{(n)}, p_{i_{n-1}+1}, x_1\right), a_{i_{n-1}+1}, \left(q_{i_{n-1}+2}^{(n)}, p_{i_{n-1}+2}, x_2\right)\right) \left(\left(q_{i_{n-1}+2}^{(n)}, p_{i_{n-1}+2}, x_2\right), a_{i_{n-1}+2}, \left(q_{i_{n-1}+3}^{(n)}, p_{i_{n-1}+3}, x_3\right)\right)\ldots$$

where for every $j \ge 1$ the choice of x_j is done as follows. We have ($x_j = 0$ and (nondeterministically) $x_{j+1} = 1$ if $q_{i_{n-1}+j+1}^{(n)} \in F_n$) or ($x_j = 1$ and $x_{j+1} = 1$ if $p_{i_{n-1}+j+1} \notin T_1$) or ($x_j = 1$ and $x_{j+1} = 2$ if $p_{i_{n-1}+j+1} \in T_1$) or ($x_j = 2$ and $x_{j+1} = 0$). Clearly, by definition of $\mathcal{C}_{1,p_{i_1+1}}, \ldots, \mathcal{C}_{n,p_{i_{n-1}+1}}$, the above paths are successful,

[5] For every $p \in P_1$, $\|\mathcal{C}_{n,p}\| = \|\mathcal{A}_n\| \odot \|\mathcal{B}_p\|$, where $\mathcal{B}_p$ is the simple cfwBa derived by $\mathcal{B}_1$ by replacing the initial distribution, with the one assigning the value 1 to p and 0 to any other state. Since, $\mathcal{A}_n, \mathcal{B}_p$ are simple, we conclude by Proposition 8 that $\|\mathcal{C}_{n,p}\|$ is simple.

[6] Abusing the definition, we call the wBa $C_{n,p}$ simple though it is not counter-free.

and we get that $weight\left(P_{w_1}^{(1)}\right)\cdot\ldots\cdot weight\left(P_{w_n}^{(n)}\right)\cdot weight(P_w) = weight\left(\overline{P}_{w_1}\right)\cdot\ldots\cdot weight\left(\overline{P}_{w_n}\right)$. Conversely, for fixed $p_{i_1+1},\ldots,p_{i_{n-1}+1}\in P_1$ such that $w\in \mathrm{supp}\left(\left\|\mathcal{C}_{1,p_{i_1+1}}\right\|\cdot\ldots\cdot\left\|\mathcal{C}_{n,p_{i_{n-1}+1}}\right\|\right)$, and successful paths $\overline{P}_{w_1},\overline{P}_{w_2},\ldots,\overline{P}_{w_n}$, we can determine the successful paths $P_{w_1}^{(1)},P_{w_2}^{(2)},\ldots,P_{w_n}^{(n)},P_w$ such that $weight\left(\overline{P}_{w_1}\right)\cdot\ldots\cdot weight\left(\overline{P}_{w_n}\right) = weight\left(P_{w_1}^{(1)}\right)\cdot\ldots\cdot weight\left(P_{w_n}^{(n)}\right)\cdot weight(P_w)$. By Lemma 1 we conclude the required equality.

Next, for the induction step, again for simplicity, we state our claim for $m=2$. Now, we consider, for every $1\leq i\leq n-1$ and $q^{(i)}\in Q_i$, the simple cfwa $\mathcal{A}_{i,q^{(i)}} = (Q_i, in_i, wt_i, \{q^{(i)}\})$ and $\mathcal{A}'_{i,q^{(i)}} = (Q_i, in'_i, wt_i, F_i)$ with $in'_i(q)=1$ if $q=q^{(i)}$, and $in'_i(q)=0$ otherwise. Moreover, for every $q^{(n)}\in Q_n$ we consider the simple cfwa $\mathcal{A}_{n,q^{(n)}} = (Q_n, in_n, wt_n, \{q^{(n)}\})$ and the simple cfwBa $\mathcal{A}'_{n,q^{(n)}} = (Q_n, in'_n, wt_n, F_n)$ with $in'_n(q)=1$ if $q=q^{(n)}$, and $in'_n(q)=0$ otherwise. Then, we get that the Hadamard product $(\|\mathcal{A}_1\|\cdot\ldots\cdot\|\mathcal{A}_n\|)\odot(\|\mathcal{B}_1\|\cdot\|\mathcal{B}_2\|)$ equals to

$$\sum_{\substack{1\leq i\leq n\\ q^{(i)}\in Q_i}}\left(\left(\left(\|\mathcal{A}_1\|\cdot\ldots\cdot\left\|\mathcal{A}_{i,q^{(i)}}\right\|\right)\odot\|\mathcal{B}_1\|\right)\cdot\left(\left(\left\|\mathcal{A}'_{i,q^{(i)}}\right\|\cdot\ldots\cdot\|\mathcal{A}_n\|\right)\odot\|\mathcal{B}_2\|\right)\right)$$

and, by induction hypothesis and Theorem 3, we are done. □

10 Closing the cycle

In this section, we prove that the class of almost simple ω-counter-free series is included in the class $\omega\text{-}ULTL(K,A)$ and we conclude the main result of our paper. For this, we shall need some preliminary matter on our weighted LTL.

For every $\varphi\in LTL(K,A)$ and $n\geq 0$ we denote by $\bigcirc^n\varphi$ the n-th repetitive application of the $\bigcirc$ operator on φ, i.e., $\bigcirc^n\varphi := \underbrace{\bigcirc(\bigcirc\ldots(\bigcirc}_{n\text{ times}}\varphi)\ldots)$, and hence $\bigcirc^0\varphi=\varphi$. Then, for every $w\in A^\omega$ we have $(\|\bigcirc^n\varphi\|, w) = (\|\varphi\|, w_{\geq n})$. The *external next depth* $exnd(\varphi)$ of a formula $\varphi\in LTL(K,A)$ is defined as follows. If $\varphi=\bigcirc\psi$, then $exnd(\varphi)=exnd(\psi)+1$. In any other case, we let $exnd(\varphi)=0$. For instance $exnd(\bigcirc(\bigcirc(\Box(\bigcirc(p_a\wedge 2)))))=2$, and if $\varphi\in LTL(K,A)$ with $exnd(\varphi)=0$, then $exnd(\bigcirc^n\varphi)=n$ for every $n\geq 0$. The following lemma is concluded in a straightforward way by the definition of $stLTL(K,A)$ formulas.

Lemma 25. *Let $\psi\in stLTL(K,A)$. Then $exnd(\psi)=0$.*

For every $n\geq 0$, we denote by $stLTL(\bigcirc,n,\wedge)$ the class of all $LTL(K,A)$ formulas of the form $\bigwedge_{0\leq j\leq m}\bigcirc^{k_j}\psi_j$ with $m\geq 0$, $\max_{0\leq j\leq m}(k_j)=n$, and $\psi_j\in stLTL(K,A)$ for every $0\leq j\leq m$. We let $stLTL(\bigcirc,\wedge)=\bigcup_{n\geq 0}stLTL(\bigcirc,n,\wedge)$. Furthermore, for every $m\geq 0$, we let U_m to be the set of all $(m+1)$-tuples of the form $((\varphi_0,k_0),(\xi_1,\varphi_1,k_1),\ldots,(\xi_m,\varphi_m,k_m))$ where $\varphi_i\in stLTL(\bigcirc,k_i,\wedge)$ and $\xi_j\in abLTL(K,A)$ for every $0\leq i\leq m$ and $1\leq j\leq m$.

Definition 12. *Let* $T = ((\varphi_0, k_0), (\xi_1, \varphi_1, k_1), \ldots, (\xi_m, \varphi_m, k_m)) \in U_m$. *For every* $w \in A^\omega$ *and* $j \geq 0$ *we define the value* $\langle T,w,j\rangle \in K$ *as follows. If* $j \leq k_0 + \ldots + k_m$, *we set* $\langle T,w,j\rangle = 0$. *Otherwise, for every* $i_1, i_2, \ldots, i_m \in \mathbb{N}$ *and* $0 \leq l \leq m$ *we define the sum* $S_l = k_0 + i_1 + k_1 + \ldots + i_l + k_l$ *with the restriction that* $S_m = j - 1$. *Then, we let*

$$\langle T,w,j\rangle = \sum_{\substack{i_1,i_2,\ldots,i_m \in \mathbb{N} \\ S_m = j-1}} \left((\|\varphi_0\|, w) \cdot \prod_{1 \leq l \leq m} \left(\prod_{0 \leq j_l < i_l} \left(\|\xi_l\|, w_{\geq S_{l-1}+j_l} \right) \cdot \left(\|\varphi_l\|, w_{\geq S_{l-1}+i_l} \right) \right) \right).$$

Note that in case $m = 0$, the restriction $S_0 = j - 1$, i.e., $k_0 = j - 1$ implies that $\langle T,w,j\rangle = 0$ for every $j > k_0 + 1$. Therefore, if $m = 0$, then $\langle T,w,j\rangle = 0$ for every $j \neq k_0 + 1$, and $\langle T,w,k_0 + 1\rangle = (\|\varphi_0\|, w)$.

Composition algorithm. Let $T_1 = ((\varphi_0, k_0), (\xi_1, \varphi_1, k_1), \ldots, (\xi_m, \varphi_m, k_m)) \in U_m$ and $T_2 = ((\psi_0, l_0), (\theta_1, \psi_1, l_1), \ldots, (\theta_n, \psi_n, l_n)) \in U_n$ with $\psi_0 = \bigwedge_{0 \leq j \leq h} \bigcirc^{p_j} \varrho_j$. We consider the formula $\varrho = \varphi_m \wedge \left(\bigwedge_{0 \leq j \leq h} \bigcirc^{k_m + p_j + 1} \varrho_j \right)$ in $stLTL(\bigcirc, k_m + l_0 + 1, \wedge)$. Then, if $m = 0$ we let

$$T = ((\varrho, k_0 + l_0 + 1), (\theta_1, \psi_1, l_1), \ldots, (\theta_n, \psi_n, l_n)),$$

otherwise we let

$$T = ((\varphi_0, k_0), (\xi_1, \varphi_1, k_1), \ldots, (\xi_m, \varrho, k_m + l_0 + 1), (\theta_1, \psi_1, l_1), \ldots, (\theta_n, \psi_n, l_n)).$$

Clearly $T \in U_{m+n}$, and we claim that

$$\langle T, w, j\rangle = \sum_{0 \leq i \leq j} (\langle T_1, w, i\rangle \cdot \langle T_2, w_{\geq i}, j - i\rangle) \tag{1}$$

for every $w \in A^\omega, j \geq 0$. Assume firstly that $m = n = 0$. If $j \neq k_0 + l_0 + 2$, then both sides of the above relation equal to 0. If $j = k_0 + l_0 + 2$, then $\langle T, w, j\rangle = (\|\varrho\|, w) = (\|\varphi_0\|, w) \cdot (\|\psi_0\|, w_{\geq k_0+1}) = \langle T_1, w, k_0 + 1\rangle \cdot \langle T_2, w_{\geq k_0+1}, j - (k_0 + 1)\rangle = \sum_{0 \leq i \leq j} (\langle T_1, w, i\rangle \cdot \langle T_2, w_{\geq i}, j - i\rangle)$.

Next, assume that $n \neq 0$ or $m \neq 0$. Then, if $j > k_0 + k_1 + \ldots + k_m + 1 + l_0 + \ldots + l_n$, we assign to $\langle T, w, j\rangle$ the sum of the products of the form

$$\left((\|\varphi_0\|, w) \cdot \prod_{1 \leq l \leq m} \left(\begin{array}{l} \prod_{0 \leq j_l < i_l} \left(\|\xi_l\|, w_{\geq S_{l-1}+j_l} \right) \\ \cdot \left(\|\varphi_l\|, w_{\geq S_{l-1}+i_l} \right) \end{array} \right) \right)$$
$$\cdot \left((\|\psi_0\|, w_{\geq S_m+1}) \cdot \prod_{1 \leq h \leq n} \left(\begin{array}{l} \prod_{0 \leq j_h < i'_h} \left(\|\theta_h\|, w_{\geq S_m+1+S'_{h-1}+j_h} \right) \\ \cdot \left(\|\psi_h\|, w_{\geq S_m+1+S'_{h-1}+i'_h} \right) \end{array} \right) \right)$$

where the sum is taken over all $i_1, \ldots, i_m, i'_1, \ldots, i'_n \in \mathbb{N}$ with $k_0 + i_1 + k_1 + \ldots + i_m + k_m + 1 + l_0 + i'_1 + l_1 + \ldots + i'_n + l_n = j - 1$.

On the other side, for every $0 \leq i \leq j$, we get the value $\langle T_1, w, i\rangle$ by summing up the products

$$(\|\varphi_0\|, w) \cdot \prod_{1\leq l\leq m} \left(\prod_{0\leq j_l<i_l} \left(\|\xi_l\|, w_{\geq S_{l-1}+j_l}\right) \cdot \left(\|\varphi_l\|, w_{\geq S_{l-1}+i_l}\right) \right) \tag{2}$$

for every $i_1, \ldots, i_m \in \mathbb{N}$ with $S_m = k_0 + i_1 + k_1 + \ldots + i_m + k_m = i - 1$. Similarly, we obtain the value $\langle T_2, w_{\geq i}, j - i\rangle$ as the sum of the products

$$(\|\psi_0\|, w_{\geq i}) \cdot \prod_{1\leq h\leq n} \left(\prod_{0\leq j_h<i'_h} \left(\|\theta_h\|, w_{\geq i+S'_{h-1}+j_h}\right) \cdot \left(\|\psi_h\|, w_{\geq i+S'_{h-1}+i'_h}\right) \right) \tag{3}$$

for every $i'_1, \ldots, i'_n \in \mathbb{N}$ with $S'_n = l_0 + i'_1 + l_1 + \ldots + i'_n + l_n = (j - i) - 1$. By a straightforward calculation in the right-hand side of (1) we conclude our claim. Finally, assume that $j \leq k_0 + k_1 + \ldots + k_m + 1 + l_0 + \ldots + l_n$. Then, $\langle T, w, j\rangle = 0$, and for every $0 \leq i \leq j$ at least one of the following is true: $i \leq k_0 + \ldots + k_m$ which implies that $\langle T_1, w, i\rangle = 0$, or $j - i \leq l_0 + \ldots + l_n$, which implies that $\langle T_2, w_{\geq i}, j - i\rangle = 0$.

In the sequel, we recall an alternative definition for star-free languages which does not involve the closure under complementation. For this, we shall need the notion of bounded synchronization delay. More precisely, let $k \geq 0$ be an integer. A prefix-free set $L \subseteq A^+$ has *bounded synchronization delay* if $uvw \in L^*$ implies $uv, w \in L^*$ for every $u, w \in A^*$ and $v \in L^k$. The least integer $k \geq 0$ satisfying the aforementioned property is called the *synchronization delay of* L.

Lemma 26. *[27] A prefix-free set of delay* 0 *is also of delay* 1.

It is well-known (cf. for instance [27, Thm. 6.3]) that the class of star-free languages over A is the smallest class of languages over A containing $\emptyset$ and $\{a\}$ for every $a \in A$, and which is closed under union, concatenation and star operation restricted to prefix-free sets with bounded synchronization delay.

For every $L, F \subseteq A^\omega$ we define the infinitary language (cf. [27]) $LUF = \{w \in A^\omega \mid w = uv$ where $u \in A^*$, $v \in F$ and $u'v \in L$ for each nonempty suffix u' of $u\}$. It should be clear that $\mathrm{supp}\,(1_L U 1_F) = LUF$, where the operation U among two series $r, s \in K\langle\langle A^\omega\rangle\rangle$, is defined for every $w \in A^\omega$, by

$$(rUs, w) = \sum_{i\geq 0} \left(\prod_{0\leq j<i} (r, w_{\geq j}) \cdot (s, w_{\geq i}) \right).$$

The two subsequent lemmas are proved in [27]. Here we present a slight modification of them and for completeness shake we state their proofs.

Lemma 27. *Let $L \subseteq A^+$ be a prefix-free set with bounded synchronization delay $k \geq 1$. Let $u \in A^*$, $v \in L^{2k}$, and $w \in Y \subseteq A^\omega$ such that*

(i) $uvw \in L^k A^\omega$, and

(ii) $u'vw \in L^{k+1}A^\omega \cup (A^\omega \backslash L^k A^\omega)$ for every suffix u' of u.

Then $uv \in L^+$.

Proof. We follow the inductive proof of Lm. 6.11 (pg. 371) in [27]. The induction is on the length of u. We let first $|u| = 0$, then $uv = v \in L^{2k}$ and since $\varepsilon \notin L$, we have $L^{2k} \subseteq L^+$. Next, assume that our claim holds for $|u| \leq n-1$ and let $|u| = n$. Condition (ii) holds for $u' = u$, and hence we get $uvw = u_1 u_2 \dots u_{k+1} r$ with $u_1, u_2, \dots, u_{k+1} \in L$ and $r \in A^\omega$. We point out the following cases.

- The word u_1 is a prefix of u. Then, $u = u_1 q$ with $q \in A^*$, and we get $uvw = u_1 u_2 \dots u_{k+1} r \Rightarrow u_1 qvw = u_1 u_2 \dots u_{k+1} r \Rightarrow qvw = u_2 \dots u_{k+1} r$. Thus, we can apply the induction hypothesis to (q, v, w). We conclude that $qv \in L^+$, and thus $uv = u_1 qv \in L^+$.
- The word uv is a prefix of $u_1 u_2 \dots u_{k+1}$. Then, $u_1 u_2 \dots u_{k+1} = uvr$ with $r \in A^*$. Since L has delay k, and $v \neq \varepsilon$ we obtain that $uv \in L^+$.
- We have $|u| < |u_1|$ and $|u_1 u_2 \dots u_{k+1}| < |uv|$. Then, $u_1 = up$ and $uv = u_1 u_2 \dots u_{k+1} q = upu_2 \dots u_{k+1} q$ for some $p, q \in A^*$, which implies that $v = pu_2 \dots u_{k+1} q$. Since L has delay k, we have $q \in L^*$. Thus, $uv = u_1 u_2 \dots u_{k+1} q$ is in L^+, as wanted.

□

Lemma 28. *Let $L \subseteq A^+$ be a prefix-free set with bounded synchronization delay $k \geq 1$ and $Y \subseteq A^\omega$. Then*

$$\big(L^+\big)\, Y = LY \cup \ldots \cup L^{2k-1}Y \cup R$$

with $R = L^k A^\omega \cap \big(\big(L^{k+1}A^\omega \cup \big(A^\omega \setminus L^k A^\omega\big)\big)\, U L^{2k} Y\big)$.

Proof. Again we follow the proof of Lm. 6.12 (pg. 372) in [27]. Let $Z = LY \cup \ldots \cup L^{2k-1}Y \cup R$. First we prove that $Z \subseteq (L^+)\, Y$. Clearly, it suffices to show that $R \subseteq (L^+)\, Y$. Let $z \in R$. Then $z \in L^k A^\omega$ and $z = uvw$ with $u \in A^*$, $v \in L^{2k}$ and $w \in Y$ with $u'vw \in L^{k+1}A^\omega \cup (A^\omega \backslash L^k A^\omega)$ for each nonempty suffix u' of u. Clearly, for $u' = \varepsilon$ it holds $vw \in L^{2k}A^\omega \subseteq L^{k+1}A^\omega \cup (A^\omega \backslash L^k A^\omega)$. By the previous lemma we get $uv \in L^+$, which implies that $uvw \in (L^+)\, Y$.

We show now the opposite inclusion. Let $z \in L^n Y$ for some $n > 0$. If $n < 2k$, then $z \in Z$. Let now $z = uvw$ with $u \in L^*$, $v \in L^{2k}$ and $w \in Y$. Clearly $z \in L^k A^\omega$. Hence, it remains to prove that $u'vw \in L^{k+1}A^\omega \cup (A^\omega \backslash L^k A^\omega)$ for each nonempty suffix u' of u. Equivalently, it suffices to prove that $u'vw \in L^k A^\omega$ implies $u'vw \in L^{k+1}A^\omega$. Suppose that $u'vw \in L^k A^\omega$. Then $u'vw = xq$ with $x \in L^k$ $(1 \leq i \leq k)$ and $q \in A^\omega$. We point out the following two cases.

- x is a proper prefix of $u'v$. Let $u = pu'$, $p \in A^*$. Since $z = pu'vw = pxq$, there is a word $s \in A^+$ such that $pxs = pu'v = uv \in L^*$. Since $x \in L^k$, we have $s \in L^+$. More precisely, since $s \neq \varepsilon$, it holds $s \in L^+$, i.e., $u'v = xs$ is in $L^{k+1}A^*$, and $u'vw$ is in $L^{k+1}A^\omega$.

- $u'v$ is a prefix of x. Then $x = u'vs$ for some $s \in A^*$. Since $v \in L^{2k}$ there exist $v_1, v_2 \in L^k$ with $v = v_1v_2$. We have $x \in L^k$ and $v_1 \in L^k$, which implies that $u'v_1 \in L^+$. Hence $u'v_1v_2w \in L^{k+1}A^\omega$, and we are done.

□

Due to the idempotency of K, the subsequent result is a straightforward conclusion from the last lemma above.

Lemma 29. *Let $L \subseteq A^+$ be a prefix-free set with bounded synchronization delay $k \geq 1$ and $Y \subseteq A^\omega$. Then*

$$1_{L^+} \cdot 1_Y = (1_L \cdot 1_Y) + \ldots + (1_{L^{2k-1}} \cdot 1_Y) + r$$

with $r = 1_{L^kA^\omega} \odot \left(1_{L^{k+1}A^\omega \cup (A^\omega \setminus L^kA^\omega)} U\, (1_{L^{2k}} \cdot 1_Y)\right)$.

Lemma 30. *Let $L \subseteq A^+$ be a star-free language. Then, there exists an integer $n > 0$ and $T_i \in U_{m_i}$ $(m_i \geq 0)$ for every $1 \leq i \leq n$, such that for every $w \in A^\omega$ and $j \geq 0$ we have $(1_L, w_{<j}) = \sum_{1 \leq i \leq n} \langle T_i, w, j\rangle$.*

Proof. We state the proof by induction on the structure of L. For the empty set the tuple $T = (0, 0) \in U_0$ satisfies our claim. Let $L = \{a\}$ for $a \in A$. We consider the tuple $T = (p_a, 0) \in U_0$ [7]. Then $S_0 = 0$ and since $S_0 = j - 1$ we get that $\langle T, w, j\rangle = 0$ for $j \neq 1$. Moreover, $\langle T, w, 1\rangle = 1$ if $w(0) = a$, and $\langle T, w, 1\rangle = 0$ otherwise. Therefore $\langle T, w, j\rangle = (1_a, w_{<j})$ for every $w \in A^\omega, j \geq 0$.

Next, assume that the induction hypothesis holds for the star-free languages $L_1, L_2 \subseteq A^+$. Then, there exist $n, m, l_i, h_k \in \mathbb{N}$, and $T_i \in U_{l_i}, T'_k \in U_{h_k}$, $(1 \leq i \leq n, 1 \leq k \leq m)$ such that for every $w \in A^\omega, j \geq 0$ we have $(1_{L_1}, w_{<j}) = \sum_{1 \leq i \leq n} \langle T_i, w, j\rangle$ and $(1_{L_2}, w_{<j}) = \sum_{1 \leq k \leq m} \langle T'_k, w, j\rangle$. Firstly, let $L = L_1 \cup L_2$. Then $(1_L, w_{<j}) = (1_{L_1} + 1_{L_2}, w_{<j}) = \sum_{1 \leq i \leq n} \langle T_i, w, j\rangle + \sum_{1 \leq k \leq m} \langle T'_k, w, j\rangle$, as wanted.

Next, let $L = L_1L_2$. Then $1_{L_1L_2} = 1_{L_1} \cdot 1_{L_2}$. For every $1 \leq i \leq n$ and $1 \leq k \leq m$ we derive from T_i, T'_k the tuple $T_{i,k} \in U_{l_i+h_k}$ by applying the *Composition*

[7] In fact we transform p_a to the equivalent $stLTL\,(\bigcirc, 0, \wedge)$ formula $1 \wedge p_a$.

algorithm. Then, we get

$$\begin{aligned}(1_{L_1}\cdot 1_{L_2}, w_{<j}) &= \sum_{0\le p\le j}\left((1_{L_1}, w_{<p})\cdot\left(1_{L_2}, (w_{\ge p})_{<j-p}\right)\right)\\ &= \sum_{0\le p\le j}\left(\sum_{1\le i\le n}\langle T_i, w, p\rangle\cdot\sum_{1\le k\le m}\langle T'_k, w_{\ge p}, j-p\rangle\right)\\ &= \sum_{0\le p\le j}\left(\sum_{1\le i\le n,1\le k\le m}(\langle T_i, w, p\rangle\cdot\langle T'_k, w_{\ge p}, j-p\rangle)\right)\\ &= \sum_{1\le i\le n,1\le k\le m}\left(\sum_{0\le p\le j}(\langle T_i, w, p\rangle\cdot\langle T'_k, w_{\ge p}, j-p\rangle)\right)\\ &= \sum_{1\le i\le n,1\le k\le m}\langle T_{i,k}, w, j\rangle\end{aligned}$$

for every $w\in A^\omega, j\ge 0$.

Finally, let L be a star-free prefix-free set with bounded synchronization delay $k\ge 0$ satisfying the induction hypothesis. By Lemma 26, it suffices to consider the case $k\ge 1$. We will prove our claim for L^+. By Lemma 29, for $Y=A^\omega$, we get

$$\begin{aligned}1_{L^+}\cdot 1_{A^\omega} =&(1_L\cdot 1_{A^\omega})+\ldots+(1_{L^{2k-1}}\cdot 1_{A^\omega})+\\ &\left(1_{L^kA^\omega}\odot\left(1_{L^{k+1}A^\omega\cup(A^\omega\setminus L^kA^\omega)}U\,(1_{L^{2k}}\cdot 1_{A^\omega})\right)\right).\end{aligned}$$

We denote $2k$ simply by p. By what we have shown above, the induction hypothesis, and same arguments with the ones used in the previous inductive step, we can prove that for every $1\le h\le p$ there exist an $n_h\in\mathbb{N}$, so that the following hold. For every $1\le i\le n_h$ there exist an $m_i\ge 0$ and a $T_{h,i}\in U_{m_i}$ with $(1_{L^h}, w_{<j}) = \sum_{1\le i\le n_h}\langle T_{h,i}, w, j\rangle$, for every $w\in A^\omega, j\ge 0$.

Let $\varphi', \widetilde{\varphi}\in bLTL(K,A)$ with semantics $1_{L^kA^\omega}$, $1_{L^{k+1}A^\omega\cup(A^\omega\setminus L^kA^\omega)}$, respectively. We set $\overline{\varphi}=\varphi'$ if $\varphi'\in stLTL(\bigcirc,0,\wedge)$ and $\overline{\varphi}=1\wedge\varphi'$, otherwise. Clearly, φ' and $1\wedge\varphi'$ are equivalent and $1\wedge\varphi'\in stLTL(\bigcirc,0,\wedge)$. We fix an $1\le i\le n_p$, and we denote for simplicity $T_{p,i}, U_{m_i}$ (where $T_{p,i}\in U_{m_i}$) with T, U_m, respectively. Let

$$T=((\psi_0,l_0),(\varphi_1,\psi_1,l_1),\ldots,(\varphi_m,\psi_m,l_m))$$

and define the tuple $T'\in U_{m+1}$ by

$$T'=((\overline{\varphi},0),(\widetilde{\varphi},\psi_0,l_0),(\varphi_1,\psi_1,l_1),\ldots,(\varphi_m,\psi_m,l_m)).$$

Then, for every $w\in A^\omega, j>l_0+\ldots+l_m$ we have

$$\langle T', w, j\rangle = \sum_{0\le q<j-(l_0+\ldots+l_m)}\left((\|\overline{\varphi}\|, w)\cdot\left(\prod_{0\le h<q}(\|\widetilde{\varphi}\|, w_{\ge h})\right)\cdot\langle T, w_{\ge q}, j-q\rangle\right) \tag{4}$$

and $\langle T', w, j\rangle = 0$ for every $j \leq l_0 + \ldots + l_m$. We repeat the same procedure for every $1 \leq i \leq n_p$ and we get the corresponding (m_i+1)-tuple $T'_{p,i}$.

Now, we show that for every $w \in A^\omega, j \geq 0$ we have

$$(1_{L^+}, w_{<j}) = \sum_{1\leq h\leq p-1} \left(\sum_{1\leq i\leq n_h} \langle T_{h,i}, w, j\rangle \right) + \sum_{1\leq i\leq n_p} \left\langle T'_{p,i}, w, j\right\rangle .$$

To this end, let $w_{<j} \in L^+$, hence either $w_{<j} \in \bigcup\limits_{1\leq h\leq p-1} L^h$ or $w_{<j} \in \bigcup\limits_{h\geq p} L^h$. In the first case $\sum\limits_{1\leq h\leq p-1} (1_{L^h}, w_{<j}) = 1$ and so $\sum\limits_{1\leq h\leq p-1} \left(\sum\limits_{1\leq i\leq n_h} \langle T_{h,i}, w, j\rangle \right) = 1$. In the latter case, $\exists u \in L^*, v \in L^p$ such that $w_{<j} = uv$. Since $v = \left(w_{\geq |u|}\right)_{<|v|}$ and $(1_{L^p}, v) = 1$, by induction hypothesis, we get that $\sum\limits_{1\leq i\leq n_p} \left\langle T_{p,i}, w_{\geq |u|}, |v|\right\rangle = \sum\limits_{1\leq i\leq n_p} \left\langle T_{p,i}, w_{\geq |u|}, j-|u|\right\rangle = 1$. Then, by the proof of Lemma 28, we get that for every suffix u' of u we have $u'vw_{\geq j} \in L^{k+1}A^\omega \cup \left(A^\omega \setminus L^kA^\omega\right)$. Hence, $(\|\overline{\varphi}\|, w) \cdot \left(\prod\limits_{0\leq h<|u|} (\|\widetilde{\varphi}\|, w_{\geq h}) \right) \cdot \left\langle T_{p,i}, w_{\geq |u|}, j-|u|\right\rangle = 1$ for some $1 \leq i \leq n_p$. By this and relation (4), we conclude that $\sum\limits_{1\leq i\leq n_p} \left\langle T'_{p,i}, w, j\right\rangle = 1$. Therefore, $(1_{L^+}, w_{<j}) = 1$ implies $\sum\limits_{1\leq h\leq p-1} \left(\sum\limits_{1\leq i\leq n_h} \langle T_{h,i}, w, j\rangle \right) = 1$ or $\sum\limits_{1\leq i\leq n_p} \left\langle T'_{p,i}, w, j\right\rangle = 1$, as required.

Conversely, assume that $\sum\limits_{1\leq h\leq p-1} \left(\sum\limits_{1\leq i\leq n_h} \langle T_{h,i}, w, j\rangle \right) = 1$ or $\sum\limits_{1\leq i\leq n_p} \left\langle T'_{p,i}, w, j\right\rangle = 1$. If the first one is true, then $\sum\limits_{1\leq h\leq p-1} (1_{L^h}, w_{<j}) = 1$. Otherwise, if the latter case holds, then there is an $1 \leq i \leq n_p$ such that $\left\langle T'_{p,i}, w, j\right\rangle = 1$. This implies that $j > l_0 + \ldots + l_{m_i}$, and by relation (4) we get

$$\left\langle T'_{p,i}, w, j\right\rangle = \sum_{0\leq q<j-(l_0+\ldots+l_{m_i})} \left((\|\overline{\varphi}\|, w) \cdot \prod_{0\leq h<q} (\|\widetilde{\varphi}\|, w_{\geq h}) \cdot \langle T_{p,i}, w_{\geq q}, j-q\rangle \right) = 1.$$

Therefore, $(\|\overline{\varphi}\|, w) = 1$, and for some $0 \leq q < j - (l_0 + \ldots + l_{m_i})$ we have $(\|\widetilde{\varphi}\|, w_{\geq h}) = \left(1_{L^{k+1}A^\omega \cup (A^\omega \setminus L^kA^\omega)}, w_{\geq h}\right) = 1$ for every $0 \leq h < q$, and $\langle T_{p,i}, w_{\geq q}, j-q\rangle = \left(1_{L^p}, (w_{\geq q})_{<j-q}\right) = 1$. We set $u = w_{<q}$, and $v = (w_{\geq q})_{<j-q}$. Then $w = uvw_{\geq j}$ and the requirements of Lemma 27 are fulfilled. We conclude that $w_{<j} = uv \in L^+$, i.e., $(1_{L^+}, w_{<j}) = 1$, and our proof is completed. □

Remark 1. By the above inductive proof, we get that for every star-free language $L \subseteq A^+$ we can find a unique integer $n > 0$ and a unique (up to formulas' equivalence) set of tuples $(T_i)_{1\leq i\leq n}$, with $T_i \in U_{m_i}$ $(m_i \geq 0)$ for every $1 \leq i \leq n$, satisfying

Lemma 30. More interestingly, we get that $\sum_{1\leq i\leq n}\langle T_i,w,j\rangle = \sum_{1\leq i\leq n}\langle T_i,w',j\rangle$ for every $w,w'\in A^\omega$ with $w_{<j}=w'_{<j}$.

Example 3. Let $A=\{a,b\}$ and $L=\{ab\}$. Clearly, L is a prefix-free set with bounded synchronization delay $k=1$. Following the inductive construction of the previous proof we get: $\varphi' = p_a \wedge \bigcirc p_b$, $\varphi_{L^2A^\omega} = p_a \wedge \bigcirc p_b \wedge \bigcirc^2 p_a \wedge \bigcirc^3 p_b$, $\varphi_{A^\omega\setminus LA^\omega} = \neg(p_a \wedge \bigcirc p_b)$ and $\widetilde{\varphi} = \varphi_{L^2A^\omega} \vee \varphi_{A^\omega\setminus LA^\omega}$. We set $T_1=(\varphi',1)$ and $T_2 = ((1\wedge\varphi',0),(\widetilde{\varphi},\varphi_{L^2A^\omega},3))$. Then, $(1_{L^+},w_{<j}) = \langle T_1,w,j\rangle + \langle T_2,w,j\rangle$ for every $w\in A^\omega$, $j\geq 0$. For instance, for every $w\in A^\omega$, $\langle T_1,w,j\rangle = 1$ iff $(j=2$ and $w_{<2}=ab)$. Let now $w=abababu$ where $u\in A^\omega$. Then,

$$\begin{aligned}\langle T_2,w,6\rangle &= \sum_{\substack{i_1\in\mathbb{N}\\ 0+i_1+3=5}} \left((\|\varphi'\|,w)\cdot \prod_{0\leq j_1<i_1} (\|\widetilde{\varphi}\|,w_{\geq j_1})\cdot(\|\varphi_{L^2A^\omega}\|,w_{\geq i_1})\right)\\ &= (\|\varphi'\|,w)\cdot(\|\widetilde{\varphi}\|,w)\cdot(\|\widetilde{\varphi}\|,w_{\geq 1})\cdot(\|\varphi_{L^2A^\omega}\|,w_{\geq 2})\\ &= 1 = (1_{L^+},w_{<6}).\end{aligned}$$

Similarly,

$$\begin{aligned}\langle T_2,w,5\rangle &= \sum_{\substack{i_1\in\mathbb{N}\\ 0+i_1+3=4}} \left((\|\varphi'\|,w)\cdot \prod_{0\leq j_1<i_1} (\|\widetilde{\varphi}\|,w_{\geq j_1})\cdot(\|\varphi_{L^2A^\omega}\|,w_{\geq i_1})\right)\\ &= (\|\varphi'\|,w)\cdot(\|\widetilde{\varphi}\|,w)\cdot(\|\varphi_{L^2A^\omega}\|,w_{\geq 1})\\ &= 0 = (1_{L^+},w_{<5}).\end{aligned}$$

It should be clear that the values obtained by the semantics of the formulas $\varphi',\widetilde{\varphi},\varphi_{L^2A^\omega}$ that appear in the computation of $\langle T_2,w,6\rangle$ do not depend on the suffix $u=w_{\geq 6}$ of w, but only on the prefix $w_{<6}$. This implies that for $w'=abababu'$ where $u'\neq u$ $(u'\in A^\omega)$ we get that $\langle T_2,w',6\rangle = \langle T_2,w,6\rangle$. A similar observation can be made for $\langle T_2,w,5\rangle$.

Example 4. Let $A=\{a,b\}$ and $L=a^+b$. For every $w\in A^\omega, j\geq 0$ it holds $(1_b,w_{<j}) = \langle T_1,w,j\rangle$ and $(1_{a^+},w_{<j}) = \langle T_2,w,j\rangle + \langle T_3,w,j\rangle$ where $T_1=(p_b,0)$, $T_2=(p_a,0)$, and $T_3 = ((p_a,0),((p_a\wedge\bigcirc p_a)\vee\neg p_a, p_a\wedge\bigcirc p_a,1))$. We apply the composition algorithm to T_3 and T_1 (resp. T_2 and T_1) and derive the tuple $T_4 = \big((p_a,0),\big((p_a\wedge\bigcirc p_a)\vee\neg p_a, p_a\wedge\bigcirc p_a\wedge\bigcirc^2 p_b,2\big)\big)$ (resp. $T_5=(p_a\wedge\bigcirc p_b,1)$). Then $(1_L,w_{<j}) = \langle T_4,w,j\rangle + \langle T_5,w,j\rangle$. Indeed, consider $w=aabu$ with $u\in A^\omega$. It holds $\langle T_5,w,3\rangle = 0$ and

$$\begin{aligned}&\langle T_4,w,3\rangle =\\ &\sum_{\substack{i_1\in\mathbb{N}\\ 0+i_1+2=2}} \left((\|p_a\|,w)\cdot \prod_{0\leq j_1<i_1} (\|(p_a\wedge\bigcirc p_a)\|,w_{\geq j_1})\cdot\big(\big\|p_a\wedge\bigcirc p_a\wedge\bigcirc^2 p_b\big\|,w_{\geq i_1}\big)\right)\\ &= (\|p_a\|,w)\cdot\big(\big\|p_a\wedge\bigcirc p_a\wedge\bigcirc^2 p_b\big\|,w\big)\\ &= 1 = (1_L,w_{<3}),\end{aligned}$$

i.e., $(1_L, w_{<3}) = 1 = \langle T_4, w, 3\rangle + \langle T_5, w, 3\rangle$, as wanted.

Proposition 9. *Let $L \subseteq A^+$ be a star-free language and $r \in K\langle\langle A^*\rangle\rangle$ a letter-step series. Then, for every $\varphi \in ULTL(K, A)$ the infinitary series $(1_L \odot r^+) \cdot \|\varphi\|$ is ω-ULTL-definable.*

Proof. Let $r = \sum_{a\in A} (k_a)_a$ where $k_a \in K$ for every $a \in A$. We set $\zeta = \bigvee_{a\in A} (k_a \wedge p_a)$. By the previous lemma there exist an $n > 0$ and $T_q \in U_{m_q}$ $(m_q \geq 0)$ for every $1 \leq q \leq n$, such that for every $w \in A^\omega, j \geq 0$ we have $(1_L, w_{<j}) = \sum\limits_{1\leq q\leq n} \langle T_q, w, j\rangle$. We fix a $1 \leq q \leq n$ and let us assume that

$$T_q = \left((\varphi_0, k_0), (\xi_1, \varphi_1, k_1), \ldots, \left(\xi_{m_q}, \varphi_{m_q}, k_{m_q}\right)\right).$$

We define the tuple $T'_q \in U_{m_q}$ by

$$T'_q = \left((\varphi'_0, k_0), (\xi'_1, \varphi'_1, k_1), \ldots, \left(\xi'_{m_q}, \varphi'_{m_q}, k_{m_q}\right)\right)$$

as follows.

- If $m_q = 0$, then $\varphi'_0 = \varphi_0 \wedge \left(\bigwedge\limits_{0\leq h\leq k_0} \bigcirc^h \zeta\right)$.
- If $m_q > 0$, then $\xi'_l = \xi_l \wedge \zeta$ for every $1 \leq l \leq m_q$. Moreover, for every $0 \leq l \leq m_q - 1$, if $k_l \neq 0$, then we let $\varphi'_l = \varphi_l \wedge \left(\bigwedge\limits_{0\leq h\leq k_l-1} \bigcirc^h \zeta\right)$, otherwise $\varphi'_l = \varphi_l$. Finally, we set $\varphi'_{m_q} = \varphi_{m_q} \wedge \left(\bigwedge\limits_{0\leq h\leq k_{m_q}} \bigcirc^h \zeta\right)$.

We show that $\left\langle T'_q, w, j\right\rangle = \langle T_q, w, j\rangle \cdot (r^+, w_{<j})$ for every $w \in A^\omega, j \geq 0$. Indeed, assume firstly that $m_q = 0$. Then, for every $j \neq k_0 + 1$ we get $\left\langle T'_q, w, j\right\rangle = \langle T_q, w, j\rangle = 0$ which implies that $\left\langle T'_q, w, j\right\rangle = \langle T_q, w, j\rangle \cdot (r^+, w_{<j})$. For $j = k_0 + 1$ we have

$$\begin{aligned}
\left\langle T'_q, w, k_0 + 1\right\rangle &= \left(\left\|\varphi_0 \wedge \left(\bigwedge_{0\leq h\leq k_0} \bigcirc^h \zeta\right)\right\|, w\right) \\
&= (\|\varphi_0\|, w) \cdot \left(\left\|\bigwedge_{0\leq h\leq k_0} \bigcirc^h \zeta\right\|, w\right) \\
&= \langle T_q, w, k_0 + 1\rangle \cdot \prod_{0\leq h\leq k_0} \left(\sum_{a\in A} (k_a)_a, w(h)\right) \\
&= \langle T_q, w, k_0 + 1\rangle \cdot \left(r^+, w_{<k_0+1}\right).
\end{aligned}$$

Next let $m_q > 0$. For every $j \leq k_0 + \ldots + k_{m_q}$ we have $\langle T'_q, w, j\rangle = \langle T_q, w, j\rangle = 0$, i.e., $\langle T'_q, w, j\rangle = \langle T_q, w, j\rangle \cdot (r^+, w_{<j})$. For every $j > k_0 + \ldots + k_{m_q}$ it holds

$$\langle T'_q, w, j\rangle = \sum_{\substack{i_1, i_2, \ldots, i_{m_q} \in \mathbb{N} \\ S_{m_q} = j-1}} \left((\|\varphi'_0\|, w) \cdot \prod_{1 \leq l \leq m_q} \left(\begin{array}{c} \prod\limits_{0 \leq j_l < i_l} \left(\|\xi'_l\|, w_{\geq S_{l-1}+j_l}\right) \\ \cdot \left(\|\varphi'_l\|, w_{\geq S_{l-1}+i_l}\right) \end{array} \right) \right).$$

By definition we have

- $(\|\varphi'_0\|, w) = (\|\varphi_0\|, w) \cdot \prod\limits_{0 \leq h \leq k_0 - 1} (r, w(h))$,
- $\left(\|\xi'_l\|, w_{\geq S_{l-1}+j_l}\right) = \left(\|\xi_l\|, w_{\geq S_{l-1}+j_l}\right) \cdot (r, w(S_{l-1} + j_l))$ for every $1 \leq l \leq m_q$ and $0 \leq j_l < i_l$,
- $\left(\|\varphi'_l\|, w_{\geq S_{l-1}+i_l}\right) = \left(\|\varphi_l\|, w_{\geq S_{l-1}+i_l}\right) \cdot \prod\limits_{0 \leq h \leq k_l - 1} (r, w(S_{l-1} + i_l + h))$ for every $1 \leq l \leq m_q - 1$, and
- $\left(\left\|\varphi'_{m_q}\right\|, w_{\geq S_{m_q-1}+i_{m_q}}\right) = \left(\|\varphi_{m_q}\|, w_{\geq S_{m_q-1}+i_{m_q}}\right) \cdot \prod\limits_{0 \leq h \leq k_{m_q}} \left(r, w\left(S_{m_q-1} + i_{m_q} + h\right)\right)$.

Hence

$$\begin{aligned}
&\langle T'_q, w, j\rangle \\
&= \sum_{\substack{i_1, i_2, \ldots, i_{m_q} \in \mathbb{N} \\ S_{m_q} = j-1}} \left(\begin{array}{c} (\|\varphi_0\|, w) \cdot \prod\limits_{1 \leq l \leq m_q} \left(\begin{array}{c} \prod\limits_{0 \leq j_l < i_l} \left(\|\xi_l\|, w_{\geq S_{l-1}+j_l}\right) \\ \cdot \left(\|\varphi_l\|, w_{\geq S_{l-1}+i_l}\right) \end{array} \right) \\ \cdot \prod\limits_{0 \leq h \leq S_{m_q}} (r, w(h)) \end{array} \right) \\
&= \sum_{\substack{i_1, i_2, \ldots, i_{m_q} \in \mathbb{N} \\ S_{m_q} = j-1}} \left((\|\varphi_0\|, w) \cdot \prod_{1 \leq l \leq m_q} \left(\begin{array}{c} \prod\limits_{0 \leq j_l < i_l} \left(\|\xi_l\|, w_{\geq S_{l-1}+j_l}\right) \\ \cdot \left(\|\varphi_l\|, w_{\geq S_{l-1}+i_l}\right) \end{array} \right) \right) \cdot \left(r^+, w_{<j}\right) \\
&= \langle T_q, w, j\rangle \cdot \left(r^+, w_{<j}\right).
\end{aligned}$$

Therefore, we get

$$\begin{aligned}
(1_L, w_{<j}) \cdot \left(r^+, w_{<j}\right) &= \left(\sum_{1 \leq q \leq n} \langle T_q, w, j\rangle \right) \cdot \left(r^+, w_{<j}\right) \\
&= \sum_{1 \leq q \leq n} \left(\langle T_q, w, j\rangle \cdot \left(r^+, w_{<j}\right) \right) \\
&= \sum_{1 \leq q \leq n} \langle T'_q, w, j\rangle .
\end{aligned}$$

For every $1 \leq q \leq n$, we define now the formula $\zeta_q \in ULTL(K, A)$ by

$$\zeta_q = \varphi'_0 \wedge \bigcirc^{k_0} \left(\xi'_1 U \left(\varphi'_1 \wedge \bigcirc^{k_1} \left(\xi'_2 U \left(\varphi'_2 \wedge \bigcirc^{k_2} \left(\ldots U \left(\varphi'_{m_q} \wedge \bigcirc^{k_{m_q}+1} \varphi \right) \right) \right) \right) \right) \right).$$

By induction on m_q, with straightforward calculations, we can show that

$$(\|\zeta_q\|, w) = \sum_{j \geq 0} \left(\langle T'_q, w, j \rangle \cdot (\|\varphi\|, w_{\geq j}) \right)$$

for every $w \in A^\omega$. Therefore, we conclude

$$\begin{aligned}
\left(\sum_{1 \leq q \leq n} (\|\zeta_q\|, w) \right) &= \sum_{1 \leq q \leq n} \left(\sum_{j \geq 0} \left(\langle T'_q, w, j \rangle \cdot (\|\varphi\|, w_{\geq j}) \right) \right) \\
&= \sum_{j \geq 0} \left(\sum_{1 \leq q \leq n} \left(\langle T'_q, w, j \rangle \cdot (\|\varphi\|, w_{\geq j}) \right) \right) \\
&= \sum_{j \geq 0} \left(\left(\sum_{1 \leq q \leq n} \langle T'_q, w, j \rangle \right) \cdot (\|\varphi\|, w_{\geq j}) \right) \\
&= \sum_{j \geq 0} \left(\left(1_L \odot r^+, w_{<j} \right) \cdot (\|\varphi\|, w_{\geq j}) \right) \\
&= \left(\left(1_L \odot r^+ \right) \cdot \|\varphi\|, w \right),
\end{aligned}$$

and our proof is completed. □

Our next result states that the almost simple ω-counter-free series are ω-$ULTL$-definable, and in fact concludes our theory.

Theorem 5. *ω-$asCF(K, A) \subseteq \omega$-$ULTL(K, A)$.*

Proof. Clearly it suffices to show that whenever $\mathcal{A}_1, \ldots, \mathcal{A}_{n-1}$ are simple cfwa and $\mathcal{A}_n$ is a simple cfwBa over A and K, then $\|\mathcal{A}_1\| \cdot \ldots \cdot \|\mathcal{A}_n\| \in \omega$-$ULTL(K, A)$. We let $r_i = \|\mathcal{A}_i\|$, and denote by k_i the initial weight $\neq 0$ and $k_a^{(i)}$ the weight $\neq 0$ of the transitions of $\mathcal{A}_i$ $(1 \leq i \leq n)$ labelled by $a \in A$. Since $\mathrm{supp}\,(r_n)$ is an ω-counter-free language it is also ω-LTL-definable hence, there is formula $\varphi \in bLTL(K, A)$ with $\|\varphi\| = 1_{\mathrm{supp}(r_n)}$. We let $\varphi_n = k_n \wedge \varphi \wedge \left(\Box \left(\bigvee_{a \in A} \left(k_a^{(n)} \wedge p_a \right) \right) \right)$ and we trivially get $r_n = \|\varphi_n\|$. By construction $\varphi_n \in ULTL(K, A)$. Furthermore, for every $1 \leq i \leq n-1$, the language $\mathrm{supp}\,(r_i) \setminus \{\varepsilon\} \subseteq A^*$ is counter-free hence, star-free. Since

$$r_i|_{A^+} = 1_{\mathrm{supp}(r_i) \setminus \{\varepsilon\}} \odot \left(k_i \left(\sum_{a \in A} \left(k_a^{(i)} \right)_a \right)^+ \right)$$

for every $1 \leq i \leq n-1$, and

$$r_{n-1}|_{A^+} \cdot r_n = k_{n-1}\left(\left(1_{\mathrm{supp}(r_{n-1})\setminus\{\varepsilon\}} \odot \left(\sum_{a\in A}\left(k_a^{(n-1)}\right)_a\right)^+\right)\cdot r_n\right),$$

by applying Proposition 9, we get that

$$\left(1_{\mathrm{supp}(r_{n-1})\setminus\{\varepsilon\}} \odot \left(\sum_{a\in A}\left(k_a^{(n-1)}\right)_a\right)^+\right)\cdot r_n \in \omega\text{-}ULTL(K,A)$$

which implies that there exists a $ULTL\,(K,A)$ formula φ_{n-1}^+ such that

$$\left(1_{\mathrm{supp}(r_{n-1})\setminus\{\varepsilon\}} \odot \left(\sum_{a\in A}\left(k_a^{(n-1)}\right)_a\right)^+\right)\cdot r_n = \left\|\varphi_{n-1}^+\right\|.$$

Hence, $r_{n-1}|_{A^+} \cdot r_n = \left\|k_{n-1} \wedge \varphi_{n-1}^+\right\|$. We let $\varphi_{n-1} = \left(k_{n-1} \wedge \varphi_{n-1}^+\right) \vee ((r_{n-1},\varepsilon) \wedge \varphi_n) \in ULTL\,(K,A)$ and we have $\|\varphi_{n-1}\| = r_{n-1}\cdot r_n$. Thus $r_{n-1}\cdot r_n \in \omega\text{-}ULTL(K,A)$. We proceed in the same way, and we show that $r_i \cdot \ldots \cdot r_n \in \omega\text{-}ULTL(K,A)$, for every $1 \leq i \leq n-2$, which concludes our proof. □

Now we are ready to state the coincidence of the classes of ω-$ULTL$-definable, ω-$wqFO$-definable, ω-star-free, and almost simple ω-counter-free series. More precisely, by Theorems 1, 2, 4, and 5 we get our main result.

Theorem 6 (Main theorem)**.**

$$\omega\text{-}ULTL\,(K,A) = \omega\text{-}wqFO(K,A) = \omega\text{-}SF(K,A) = \omega\text{-}asCF(K,A).$$

Conclusion

We showed the coincidence of the classes of series definable in a fragment of the weighted *LTL*, series definable in a fragment of the weighted *FO* logic, ω-star-free series, and almost simple ω-counter-free series. Our underlying semiring required to be idempotent, zero-divisor free and totally commutative complete satisfying an additional property. It is an open problem whether we can relax the idempotency and/or the zero-divisor freeness property of the semiring. Our results can be proved for series over finite words. In this case we do not need completeness axioms anymore. As a future research we state two main directions. The first one is the development of our theory in the probabilistic setup, i.e., to investigate the expressive equivalence (of fragments) of probabilistic *LTL*, probabilistic *FO* logic, probabilistic ω-star-free expressions, and counter-free probabilistic Büchi automata, where the last two concepts have not been defined yet. The latter concerns the development of our theory in the setup of more general structures than semirings. For instance, in [12] the authors studied weighted automata and weighted *MSO* logics over valuation monoids which capture operations that play an important role in practical applications.

References

[1] M. Akian, S. Gaubert, A. Guterman, Linear independence over tropical semirings and beyond, *Contemp. Math.* 495(2009) 1–38.

[2] R. Balbes, P. Dwinger, *Distributive Lattices*, University of Missouri Press, 1974.

[3] C. Baier, J.-P. Katoen, *Principles of Model Checking,* The MIT Press, 2008.

[4] B. Bollig, P. Gastin, B. Mommege, M. Zeitoun, Pebble weighted automata and transitive closure logics, in: *Proceedings of ICALP 2010, LNCS* 6199(2010) 587–598.

[5] J.R. Büchi, Weak second-order arithmetic and finite automata, *Z. Math. Logik Grundlager Math.* 6(1960) 66-92.

[6] J.R. Büchi, On a decision method in restricted second order arithmetic, in: *Proc. 1960 Int. Congr. for Logic, Methodology and Philosophy of Science,* (1962), pp.1-11.

[7] V. Diekert, P. Gastin, First-order definable languages, in: *Logic and Automata: History and Perspectives.* Texts in Logic and Games 2, Amsterdam University Press 2007, pp. 261–306.

[8] M. Droste, P. Gastin, Weighted automata and weighted logics, *Theoret. Comput. Sci.* 380 (2007) 69–86. Extended abstract in: *Proceedings of ICALP 2005, LNCS* 3580 (2005) 513–525.

[9] M. Droste, P. Gastin, Weighted automata and weighted logics, in: M. Droste, W. Kuich, H. Vogler (Eds), *Handbook of Weighted Automata*, Springer-Verlag 2009, chapter 5.

[10] M. Droste, W. Kuich, H. Vogler (eds), *Handbook of Weighted Automata.* EATCS Monographs in Theoretical Computer Science, Springer, Berlin, 2009.

[11] M. Droste, D. Kuske, Skew and infinitary formal power series, *Theoret. Comput. Sci.* 366(2006) 199–227.

[12] M. Droste, I. Meinecke, Weighted automata and weighted MSO logics for average and long-time behaviors, *Inform. and Comput.* 220–221(2012) 44–59.

[13] M. Droste, G. Rahonis, Weighted automata and weighted logics on infinite words, *Russian Math.* 54(2010) 26–45, in Russian: *Iz. VUZ* 1(2010) 34–58.

[14] M. Droste, H. Vogler, Weighted automata and multi-valued logics over arbitrary bounded lattices, *Theoret. Comput. Sci.* 418(2012) 14–36.

[15] S. Eilenberg, *Automata, Languages and Machines, vol. A,* Academic Press 1974.

[16] C. Elgot, Decision problems of finite automata design and related arithmetics, *Trans. Amer. Math. Soc.* 98(1961) 21-52.

[17] Z. Ésik, W. Kuich, A semiring-semimodule generalization of ω-regular languages I. Special issue on "Weighted automata" (M. Droste, H. Vogler, eds.) *J. of Automata Languages and Combinatorics* 10(2005) 203-242.

[18] Z. Ésik, W. Kuich, On iteration semiring-semimodule pairs, *Semigroup Forum*, 75(2007) 129–159.

[19] W. Kuich, Semirings and formal power series: Their relevance to formal languages and automata theory. In: *Handbook of Formal Languages* (G. Rozenberg, A. Salomaa, eds.), vol. 1, Springer, 1997, pp. 609–677.

[20] W. Kuich, G. Rahonis, Fuzzy regular languages over finite and infinite words, *Fuzzy Sets and Systems,* 157(2006) 1532-1549.

[21] O. Kupferman, Y. Lustig, Lattice automata, in: *Proceedings of VMCAI 2007, LNCS* 4349(2007) 199–213.

[22] O. Kupferman, A. Pnueli, M.Y. Vardi, Once and for all, *J. Comput. Syst. Sci.* 78(2012) 981–996.

[23] R.E. Ladner, Application of model-theoretic games to discrete linear orders and finite automata, *Inform. and Control* 33(1977) 281–303.

[24] E. Mandrali, *Weighted Computability with Discounting*, PhD Thesis, Aristotle University of Thessaloniki, Thessaloniki 2013.

[25] E. Mandrali, G. Rahonis, Characterizations of weighted first-order logics over semirings, in: *Proceedings of CAI 2013, LNCS* 8080(2013) 247–259.

[26] E. Mandrali, G. Rahonis, On weighted first-order logics with discounting, *Acta Inform.* 51(2014) 61–106.

[27] D. Perrin, J.-É. Pin, *Infinite Words,* Pure and Applied Mathematics, Elsevier, 2004.

[28] J.-É. Pin, Logic on words, in: *Current Trends in Theoretical Computer Science.* World Sci. Publ., River Edge, NJ, 2001, pp.254–273.

[29] W. Thomas, Star-free regular sets of ω-sequences, *Inform. and Control* 42(1979) 148–156.

[30] W. Thomas, Languages, automata and logic, in: *Handbook of Formal Languages* vol. 3 (G. Rozenberg, A. Salomaa, eds.), Springer, 1997, pp. 389-485.

[31] M.Y. Vardi, From philosophical to industrial logics, in: *Proceedings of ICLA 2009, LNAI* 5378(2009) 89–115.

Received 20th June 2015

Acta Cybernetica **22** (2015) 485–498.

On Shift Radix Systems over Imaginary Quadratic Euclidean Domains*

Attila Pethő† Péter Varga‡ and Mario Weitzer§

Dedicated to the memory of Professor Ferenc Gécseg

Abstract

In this paper we generalize the shift radix systems to finite dimensional Hermitian vector spaces. Here the integer lattice is replaced by the direct sum of imaginary quadratic Euclidean domains. We prove in two cases that the set of one dimensional Euclidean shift radix systems with finiteness property is contained in a circle of radius 0.99 around the origin. Thus their structure is much simpler than the structure of analogous sets.

1 Introduction

For $\mathbf{r} \in \mathbb{R}^n$ the mapping $\tau_{\mathbf{r}} : \mathbb{Z}^n \mapsto \mathbb{Z}^n$, defined as

$$\tau_{\mathbf{r}}((a_1, \ldots, a_n)) = (a_2, \ldots, a_{n-1}, \lfloor \mathbf{ra} \rfloor),$$

where $\mathbf{ra}$ denotes the inner product, is called *shift radix system*, shortly SRS. This concept was introduced by S. Akiyama et al. [1] and they proved that it is a common generalization of canonical number systems (CNS), first studied by I. Kátai and J. Szabó [8], and the β-expansions, defined by A. Rényi [11]. For computational aspects of CNS we refer to the paper of P. Burcsi and A. Kovács [5].

Among the several generalizations of CNS we cite here only one to polynomials over Gaussian integers by M.A. Jacob and J.P. Reveilles [7]. Generalizing the shift radix systems, H. Brunotte, P. Kirschenhofer and J. Thuswaldner [3] defined GSRS for Hermitian vector spaces. A wider generalization of CNS, namely for polynomials over imaginary quadratic Euclidean domains was studied by the first two authors

*Research supported in part by the OTKA grants NK104208, NK101680. Mario Weitzer is supported by the Austrian Science Fund (FWF): W1230, Doctoral Program "Discrete Mathematics".

†University of Debrecen, Department of Computer Science, H–4010 Debrecen P.O. Box 12, Hungary. University of Ostrava, Faculty of Science, Dvořákova 7, 70103 Ostrava, Czech Republik. E-mail: petho.attila@inf.unideb.hu

‡H–4031 Debrecen, Gyepűsor utca 12., Hungary. E-mail: vapeti@gmail.com

§Chair of Mathematics and Statistics, University of Leoben, Franz-Josef-Strasse 18, A-8700 Leoben, Austria. E-mail: mario.weitzer@unileoben.ac.at

in [10]. It is well known that there are exactly five such domains, which are the ring of integers of the imaginary quadratic fields $\mathbb{Q}(\sqrt{d}), d = 1, 2, 3, 7, 11$. The Euclidean norm function allows not only the division by remainder, but also to define a floor function for complex numbers. This observation leads us to generalize SRS for Hermitian vector spaces endowed by floor functions depending on imaginary quadratic Euclidean domains. Our generalization, which we call ESRS, is uniform for the five imaginary quadratic Euclidean domains. This has the consequence that in case of the Gaussian integers our floor function differs from that used in [3].

The SRS $\tau_{\mathbf{r}}$ is said to have the finiteness property iff for all $\mathbf{a} \in \mathbb{Z}^{\mathbf{n}}$ there exists a $k \geq 1$ such that $\tau_{\mathbf{r}}^k(\mathbf{a}) = \mathbf{0}$. Denote by $\mathcal{D}_n^{(0)}$ the set of $\mathbf{r} \in \mathbb{R}^n$ such that $\tau_{\mathbf{r}}$ has the finiteness property. From numeration point of view these real vectors are most important. It turned out that the structure of $\mathcal{D}_n^{(0)}$ is very complicated already for $n = 2$, see [2], [12] and [13].

The analogue of the two dimensional SRS is the one dimensional GSRS and ESRS. Brunotte et al. [3] studied first the set of one dimensional GSRS with finiteness property, which we denote by $\mathrm{GSRS}^{(0)}$. It turned out that its structure is quite complicated as well. Recently a more precise investigation of M. Weitzer [14] showed that the structure of $GSRS^{(0)}$ is much simpler as that of $\mathcal{D}_2^{(0)}$. Based on extensive computer investigations he conjectures a finite description of $\mathrm{GSRS}^{(0)}$.

Analogously to $\mathcal{D}_n^{(0)}$ we can define $\mathcal{D}_{n,d}^{(0)}, d = 1, 2, 3, 7, 11$ in a straight forward way. We show how one can compute good approximations of $\mathcal{D}_{n,d}^{(0)}$. Performing the computation it turned out that the shape of these objects are quite different. The subjective impression can be misleading, but we were able to prove that $\mathcal{D}_{n,d}^{(0)}$ has no critical points in the cases $d = 2, 11$. More specifically we prove that the circle of radius 0.99 around the origin contains $\mathcal{D}_{n,d}^{(0)}$. In the other cases this is probably not true. It is certainly not true for $\mathcal{D}_2^{(0)}$ and $\mathrm{GSRS}^{(0)}$.

2 Basic concepts

In order to establish a shift radix system over the complex numbers, an imaginary quadratic Euclidean domain will be used as the set of integers, and a floor function is needed which can be determined by making its Euclidean function unique, so choosing the set of fractional numbers from the possible values.

Definition 1. *Let $\mathbb{E}_d = \mathbb{Z}_{\mathbb{Q}[\sqrt{-d}]}$ be an imaginary quadratic Euclidean domain ($d \in \{1, 2, 3, 7, 11\}$, see in [6]). Its **canonical integral basis** is: $\{1, \omega\}$, where*

$$\omega := \begin{cases} \sqrt{-d} & , \text{ if } d \in \{1, 2\}, \\ \frac{1+\sqrt{-d}}{2} & , \text{ otherwise.} \end{cases}$$

(In the case of $d = 1$ instead of ω the imaginary unit i is used.)

For fixed d, the complex numbers $1, \omega$ form a basis of $\mathbb{C}$, as a two dimensional vector space over $\mathbb{R}$. Thus all $z \in \mathbb{C}$ can be uniquely written in the form $z = e_1 + e_2\omega$

with $e_1, e_2 \in \mathbb{R}$. Plainly $z \in \mathbb{E}_d$ iff $e_1, e_2 \in \mathbb{Z}$. Let the functions $Re_d : \mathbb{C} \mapsto \mathbb{R}$ and $Im_d : \mathbb{C} \mapsto \mathbb{R}$ be defined as:

$$Re_d(z) := e_1, \ Im_d(z) := e_2.$$

$Re_d(z)$ and $Im_d(z)$ are called the *real and imaginary parts of* z.
The elements of $\mathbb{E}_d$ will be denoted by $(e_1, e_2)_d$.

Plainly, for all $z \in \mathbb{C}$ we have

$$\begin{aligned} Im_d(z) &= \frac{Im(z)}{Im(\omega)}, \\ Re_d(z) &= Re(z) - Im(z)\frac{Re(\omega)}{Im(\omega)}. \end{aligned}$$

In order to define a floor function, a set of fractional numbers has to be defined. Regarding generalization purposes the absolute value of a fractional number should be less than 1, a fractional number should not be negative in a sense, it is a superset of the fractional numbers for the reals, and the floor function should be unambiguous. From these considerations the following definition will be used to specify the floor function with the set of fractional numbers which will be called fundamental sail tile.

Definition 2. *Let* $d \in \{1, 2, 3, 7, 11\}$. *Let the set*

$$\mathbb{D}_d := \left\{ c \in \mathbb{C} \ \middle| \ |c| < 1 \ \ |c+1| \geq 1 \ \ -\frac{1}{2} \leq Im_d(c) < \frac{1}{2} \right\}$$

be defined as the ***fundamental sail tile*** *(the set of fractional numbers).*
Let $p \in \mathbb{E}_d$. *The set*

$$\mathbb{D}_d(p) := \left\{ p + c \ \middle| \ c \in \mathbb{C} \ \ |c| < 1 \ \ |c+1| \geq 1 \ \ -\frac{1}{2} \leq Im_d(c) < \frac{1}{2} \right\}$$

is called p***-sail tile*** *and* p *is called its* ***representative integer****.*

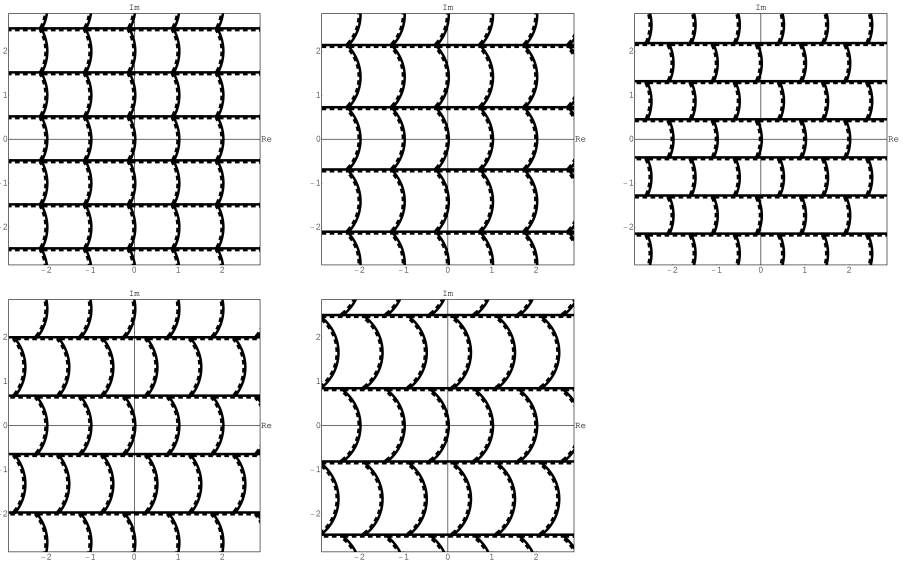

Figure 1: Tilings of $\mathbb{C}$ given by the sets $\mathbb{D}_d(p)$, $d \in \{1, 2, 3, 7, 11\}$.

By using Theorem 1 of [10] one can show that the sets $\mathbb{D}_d(p)$, where p runs through $\mathbb{E}_d$ do not overlap and cover the complex plain $\mathbb{C}$. This justifies the following definition:

Definition 3. *Let the function* $\lfloor\ \rfloor_d : \mathbb{C} \to \mathbb{E}_d$ *be defined as the* ***floor function****. The floor of* e *is the representative integer* p *of the unique* p*-sail tile that contains* e*.*

The next lemma shows that the above defined floor function can be described with the well-known floor function over the real numbers. We leave its simple proof to the reader.

Lemma 1.

$$\lfloor e \rfloor_d = \begin{cases} \lfloor Re(e) - \lfloor Im_d(e) + \frac{1}{2} \rfloor Re(\omega) \rfloor + \omega \lfloor Im_d(e) + \frac{1}{2} \rfloor, \text{ if} \\ \quad \Big(Re(e) - \lfloor Re(e) - \lfloor Im_d(e) + \frac{1}{2} \rfloor Re(\omega) \rfloor - \\ \quad\quad - \lfloor Im_d(e) + \frac{1}{2} \rfloor Re(\omega) \Big)^2 + \\ \quad + \big(Im(e) - \lfloor Im_d(e) + \frac{1}{2} \rfloor Im(\omega) \big)^2 < 1, \\ \lfloor Re(e) - \lfloor Im_d(e) + \frac{1}{2} \rfloor Re(\omega) \rfloor + \omega \lfloor Im_d(e) + \frac{1}{2} + 1 \rfloor, \text{ otherwise.} \end{cases}$$

Equipped with the appropriate floor functions we are in the position to define

shift radix systems for Hermitian vectors. The notion depends on the imaginary Euclidean domain.

Definition 4. *Let $C := (c_1, \ldots, c_n) \in \mathbb{C}^n$ be a complex vector. Let $d \in \{1, 2, 3, 7, 11\}$ and the floor function $\lfloor x \rfloor_d$ defined as above.*
For all vectors $A := (a_1, a_2, \ldots, a_n) \in \mathbb{E}_d^n$ let

$$\tau_{d,C}(A) := (a_2, \ldots, a_n, -q)\,,$$

where $q = \lfloor c_1 a_1 + c_2 a_2 + \cdots + c_n a_n \rfloor_d$. The mapping $\tau_{d,C} : \mathbb{E}_d^n \mapsto \mathbb{E}_d^n$ is called ***Euclidean shift radix system with parameter*** *d or $ESRS_d$ respectively, ESRS for short. If $B := \tau_{d,C}(A)$, this mapping will be denoted by*

$$A \underset{d,C}{\Rightarrow} B.$$

If for $A, B \in \mathbb{E}_d^n$ there is a $k \in \mathbb{N}$, such that $\tau_{d,C}^k(A) = B$ then this will be indicated by:

$$A \underset{d,C}{\overset{*}{\Longrightarrow}} B.$$

$\tau_{d,C}$ is called ***ESRS with finiteness property*** *iff for all vectors $A \in \mathbb{E}_d^n$*

$$A \underset{d,C}{\overset{*}{\Longrightarrow}} 0,$$

where 0 is the zero vector.

Definition 5. *The following sets form a generalization of the corresponding sets defined in [1]:*

$$\mathcal{D}_{n,d}^0 := \left\{ C \in \mathbb{C}^n \;\middle|\; \forall A \in \mathbb{E}_d^n : A \underset{d,C}{\overset{*}{\Longrightarrow}} 0 \right\},$$

$$\mathcal{D}_{n,d} := \left\{ C \in \mathbb{C}^n \;\middle|\; \forall A \in \mathbb{E}_d^n \text{ the sequence } \left\{ \tau_{d,C}^k(A) \right\}_{k \geq 0} \text{ is ultimately periodic} \right\}.$$

$\tau_{d,C}$ is ESRS with finiteness property iff $C \in \mathcal{D}_{n,d}^0$.

Remark 1. *The construction defined in this section can be generalized by using a complex number for d.*

3 Basic properties of the one dimensional shift radix systems

This section and the following ones will consider C as a one dimensional vector, i.e. a complex number, which will be denoted by c. In this section we will investigate

some properties of the one dimensional case. Theorem 1 can be considered as the generalization of cutout polyhedra defined in [1]. These are areas defined by a closed curve (arcs and lines). Let this area be denoted by P. Let's consider this as **cutout area**.

Theorem 1. *Let $c \in \mathbb{C}$. The number $a_0 \in \mathbb{E}_d$ with (d, c) admits a period*

$$a_0 \underset{d,c}{\Rightarrow} a_1 \underset{d,c}{\Rightarrow} a_2 \underset{d,c}{\Rightarrow} a_3 \ldots \underset{d,c}{\Rightarrow} a_{l-1} \underset{d,c}{\Rightarrow} a_0, \text{ if and only if}$$

$$c \in \left(\frac{\mathbb{D}_d - a_1}{a_0}\right) \cap \left(\frac{\mathbb{D}_d - a_2}{a_1}\right) \cap \cdots \cap \left(\frac{\mathbb{D}_d - a_{l-1}}{a_{l-2}}\right) \cap \left(\frac{\mathbb{D}_d - a_0}{a_{l-1}}\right).$$

The number l will be called the length of the period.

Proof. The proof is essentially the same as the proof of Theorem 3 in [10]. □

The next theorem shows that if the ESRS associated to c has the finiteness property then it must lie in the closed unit circle.

Theorem 2. *Let $|c| > 1$, $d \in \{1, 2, 3, 7, 11\}$ then $\tau_{d,c}$ doesn't have the finiteness property.*

Proof. The basic idea is that we ignore those values of a where the length decreases after applying $\tau_{d,c}$, since after finitely many steps it will end in 0 or another value a' the absolute value of which increases by applying the mapping. Investigating the length of a vector after applying the shift radix mapping:

$$a \underset{d,c}{\Rightarrow} ac - r.$$

For the length

$$|a| > |ac - r| \geq |a||c| - |r| > |a||c| - 1,$$

$$|a| < \frac{1}{|c| - 1}.$$

If this inequality holds the length decreases. This is a finite open disk around the origin. For any other a the length will increase, so starting from a applying the shift radix mapping leads to a divergent sequence. □

Plainly $\tau_{d,1}$ doesn't have the finiteness property for any d. For finding ESRS with finiteness property, one has to use a well chosen complex number c. Based on Theorem 2, let's start from the closed unit disc around the origin, and let's ignore these cutout areas in order to reach those points which are good to define ESRS with finiteness property:

Remark 2. *The set $\mathcal{D}^0_{n,d}$ can be defined in the following way. Let $S := \{c \in \mathbb{C} |\ |c| \leq 1\}$ and let's consider the areas defined by Theorem 1 as P_i. Then*

$$\mathcal{D}^0_{n,d} = S \setminus \cup P_i.$$

Since cutout areas can be infinitely many, can be disjoint, overlapped by each other or superset and subset of each other, finding the union area of all is a hard problem. The following definition helps to estimate how many cutout areas are around some point in $\mathcal{D}_{n,d}$.

Definition 6. *Let $c \in \mathcal{D}_{n,d}$.*

- *If there exists an open neighborhood of c which contains only finitely many cutout areas then we call c a **regular point**.*
- *If each open neighborhood of c has nonempty intersection with infinitely many cutout areas then we call c a **weak critical point** for $\mathcal{D}_{n,d}$.*
- *If for each open neighborhood U of c the set $U \setminus \mathcal{D}^0_{n,d}$ cannot be covered by finitely many cutout areas then c is called a **critical point**.*

Let's check what are the conditions to reach a cutout area in the one dimensional case.

Remark 3. *Theorem 1's result for one dimensional case can be used to define cutout areas with periods of any length. $\mathcal{T}_{d,c}$ admits a period $a_0 \underset{d,c}{\Rightarrow} a_1 \underset{d,c}{\Rightarrow} a_2 \underset{d,c}{\Rightarrow} \ldots \underset{d,c}{\Rightarrow} a_n \underset{d,c}{\Rightarrow} a_0$ if and only if*

$$c \in \left(\frac{\mathbb{D}_d - a_1}{a_0}\right) \cap \left(\frac{\mathbb{D}_d - a_2}{a_1}\right) \cap \cdots \cap \left(\frac{\mathbb{D}_d - a_n}{a_{n-1}}\right) \cap \left(\frac{\mathbb{D}_d - a_0}{a_n}\right).$$

The one-step and the two-step cases are really important, since the one-step periods define large sets around -1, and the two-step case appear most likely around 1. The following two lemmata speak about these special cases.

Lemma 2. *Let $|c| < 1$. $\mathcal{T}_{d,c}$ admits a one-step period, if and only if $c \in \frac{\mathbb{D}_d}{a} - 1$ for an $a \in \mathbb{E}_d \setminus \{0\}$.*

Proof. The shift radix mapping leads to the following:

$$a \underset{d,c}{\Rightarrow} -ac + r,$$

$a \in \mathbb{E}_d \setminus \{0\}$. This can be a one-step period, iff $c = \frac{r}{a} - 1$. r is a general element of the fundamental sail tile, so $c \in \frac{\mathbb{D}_d}{a} - 1$. □

Lemma 3. *Let $|c| < 1$. $\mathcal{T}_{d,c}$ admits a two-step period, if and only if $c \in \left(\frac{\mathbb{D}_d - a'}{a}\right) \cap \left(\frac{\mathbb{D}_d - a}{a'}\right)$, where $a, a' \in \mathbb{E}_d \setminus \{0\}$.*

Proof. The shift radix mapping leads to the following:

$$a \underset{d,c}{\Rightarrow} -ac + r,$$

$a \in \mathbb{E}_d \setminus \{0\}$. Let $a' := -ac + r \in \mathbb{E}_d \setminus \{0\}$, $a' \underset{d,c}{\Rightarrow} -a'c + r$. This can be a two-step period, iff $a = -a'c + r$. This means that c has to be in the set

$$c \in \left(\frac{\mathbb{D}_d - a'}{a}\right) \cap \left(\frac{\mathbb{D}_d - a}{a'}\right).$$

□

Theorem 3 shows that only finitely many $a \in \mathbb{E}_d$ have to be investigated to decide the finiteness property of a specific value of c.

Theorem 3. *Let $|c| < 1$. $\mathcal{T}_{d,c}$ is a ESRS with finiteness property, iff for all $a \in \mathbb{E}_d$ where $|a| < \frac{1}{1-|c|}$*

$$a \underset{d,c}{\overset{*}{\Rightarrow}} 0.$$

Proof.

$$a \underset{d,c}{\Rightarrow} -ac + r, \text{ where}$$

$r \in \mathbb{D}_d$. To decide the finiteness property one has to check only those numbers where the absolute value does not decrease.

$$|a| \leq |-ac + r| \leq |a||c| + |r| < |a||c| + 1, \text{ so}$$

$|a| < \frac{1}{1-|c|}$. □

Now, let's see how the sets $\mathcal{D}^0_{1,d}$ ($d \in \{1, 2, 3, 7, 11\}$) look like. Algorithm 1 defines a searching method, which will approximate the mentioned set using the results of Remark 2 and Theorem 3. The input parameters are $d \in \{1, 2, 3, 7, 11\}$ and rs, which sets how many points in the unit circle will be tested, the result is a superset of $\mathcal{D}^0_{1,d}$.

Algorithm 1 Approximation algorithm for the set $\mathcal{D}^0_{1,d}$

$d \in \{1, 2, 3, 7, 11\}$ (input parameter)
$rs := 1000000$ (input parameter)
$res := \frac{1}{\sqrt{rs}}$
$S := \{c \in \mathbb{C} \mid |c| \leq 1\}$
$S_{curr} := S$
for $rad \in \{0, res, 2res \ldots, 1\}$ **do**
    for $ang \in \{0, res, 2res \ldots, 2\pi\}$ **do**
        $c_{curr} := rad \cdot e^{i \cdot ang}$
        if $c_{curr} \in S_{curr}$ **then**
            $A_{curr} := \{a' | a' \in \mathbb{E}_d \;\; |a'| < \frac{1}{1-|c_{curr}|}\}$
            for $a_{curr} \in A_{curr}$ **do**
                if $\mathcal{T}_{d,c_{curr}}$ admits a period P' starting from a_{curr} **then**
                    $S_{curr} = S_{curr} \setminus P'$
                    break operation *11*
                end if
            end for
        end if
    end for
end for
return S_{curr}

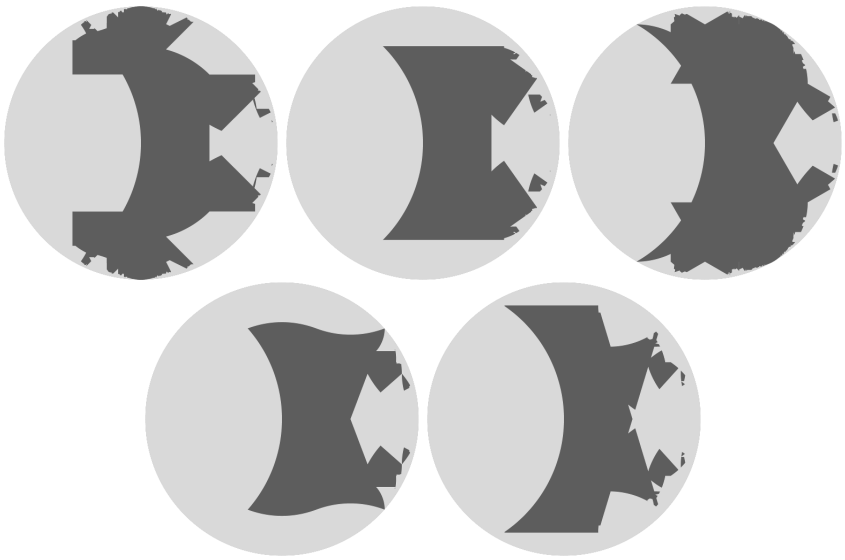

Figure 2: Using Algorithm 1, these are the generated approximations of $\mathcal{D}^0_{1,1}, \mathcal{D}^0_{1,2}, \mathcal{D}^0_{1,3}, \mathcal{D}^0_{1,7}, \mathcal{D}^0_{1,11}$, respectively (black area).

The area close to the origin is the easiest part of the disc to decide the finiteness property, so let's consider the case $|c| < \frac{1}{2}$.

Theorem 4. *Let $|c| < 1 - \frac{1}{\sqrt{4}} = \frac{1}{2}$. The function $\mathcal{T}_{d,c}$ is a ESRS with finiteness property, if $c \in \mathbb{D}_d$. Additionally, if $d = 11$ then*

$$c \notin \left\{ z \in \mathbb{C} \;\middle|\; |(-\omega)z + \omega - 1| \geq 1 \quad - \frac{\sqrt{11}}{4} < Im((-\omega)z + \omega) \right\}, \text{ and}$$

$$c \notin \left\{ z \in \mathbb{C} \;\middle|\; |(-1+\omega)z - \omega| \geq 1 \;\; Im((-1+\omega)z + 1 - \omega) \leq \frac{\sqrt{11}}{4} \right\}.$$

Proof. The proof of this theorem only uses basic considerations and the results of this article. □

The following Lemma implies that $\mathcal{D}^0_{1,d}$ and $\mathcal{D}_{1,d}$ reflected at the real axis coincide almost everywhere. Parts where the two sets might not coincide are contained in the union of their respective boundaries.

Lemma 4. *Let $c \in \mathbb{C}$, $a, b \in \mathbb{E}_d$, and $\varphi = (a_1, a_2, \ldots, a_k) \in \mathbb{E}_d^k$. Then $2Im_d(ca)$ is not an odd integer $\Leftrightarrow$ $(\mathcal{T}_c a = b \Leftrightarrow \mathcal{T}_{\overline{c}} \overline{a} = \overline{b})$,*
$2Im_d(ca)$ is an odd integer $\Rightarrow$ $(\mathcal{T}_c a = b \Rightarrow \mathcal{T}_{\overline{c}} \overline{a} - \overline{b} \in \{(0,-1)_d, (1,-1)_d\})$.
In particular, if c is contained in the interior of the cutout area corresponding to φ then
$(a_1, a_2, \ldots, a_k)$ period of $\mathcal{T}_c \Leftrightarrow (\overline{a_1}, \overline{a_2}, \ldots, \overline{a_k})$ period of $\mathcal{T}_{\overline{c}}$.

Proof. The proof can be done the same way as the proof of Lemma 3.6 in [3]. □

Definition 7. *Let*

$$\begin{aligned}
&\big(((x_{2,1}, y_{2,1}), (a_{2,1}, b_{2,1})), \ldots, ((x_{2,45}, y_{2,45}), (a_{2,45}, b_{2,45}))\big) := \Big(\\
&\quad \big((1,0), (-2,0)\big), \big((-\tfrac{529}{4023}, \tfrac{22378908}{45415717}), (0,1)\big), \quad \big((-\tfrac{8413}{3862276}, \tfrac{6385}{8993}), (0,1)\big), \\
&\quad \big((-\tfrac{560}{3763}, \tfrac{166}{229}), (0,1)\big), \big((\tfrac{11051}{36427}, \tfrac{12022}{16987}), (0,1)\big), \big((-\tfrac{39833}{139318}, \tfrac{634841}{887952}), (0,1)\big), \\
&\quad \big((-\tfrac{587}{32542}, \tfrac{1260970}{1501501}), (0,1)\big), \big((\tfrac{20911}{27059}, \tfrac{183}{517}), (0,1)\big), \big((-\tfrac{3533}{7022}, \tfrac{1411}{1988}), (0,1)\big), \\
&\quad \big((\tfrac{645}{3757}, \tfrac{1432877}{1660169}), (0,1)\big), \quad \big((\tfrac{844688}{1266909}, \tfrac{2031}{3445}), (0,4)\big), \quad \big((\tfrac{44399}{51256}, \tfrac{4447}{14348}), (0,2)\big), \\
&\quad \big((\tfrac{781981}{1137704}, \tfrac{159}{260}), (0,4)\big), \\
&\big((\tfrac{3741}{6160}, \tfrac{2237}{3237}), (0,2)\big), \quad \big((\tfrac{18563}{132052}, \tfrac{677269}{744909}), (0,1)\big), \quad \big((-\tfrac{273}{461}, \tfrac{256}{357}), (0,1)\big), \\
&\big((-\tfrac{23531}{44649}, \tfrac{2367}{3041}), (0,1)\big), \big((-\tfrac{2504}{4903}, \tfrac{53361}{66614}), (0,1)\big), \big((\tfrac{2295978}{14352937}, \tfrac{128937}{134770}), (0,1)\big), \\
&\big((-\tfrac{22537}{155137}, \tfrac{19631}{20469}), (0,1)\big), \big((-\tfrac{1324}{2503}, \tfrac{85287}{104894}), (0,1)\big), \big((\tfrac{186647}{247677}, \tfrac{278}{433}), (0,2)\big), \\
&\big((\tfrac{81473}{111068}, \tfrac{86419}{129984}), (0,2)\big), \big((-\tfrac{1087}{2004}, \tfrac{670}{809}), (0,1)\big), \big((\tfrac{19}{25}, \tfrac{16}{25}), (0,2)\big), \big((\tfrac{27}{37}, \tfrac{25}{37}), \\
&(0,2)\big), \quad \big((\tfrac{13}{17}, \tfrac{54}{85}), (0,2)\big), \quad \big((\tfrac{7647}{10000}, \tfrac{16}{25}), (0,2)\big), \quad \big((\tfrac{7339}{10000}, \tfrac{1347}{2000}), (0,2)\big), \\
&\big((\tfrac{1979}{20000}, \tfrac{4961}{5000}), (0,1)\big), \quad \big((-\tfrac{1979}{20000}, \tfrac{397}{400}), (0,1)\big), \quad \big((-\tfrac{2701}{5000}, \tfrac{8399}{10000}), (0,1)\big), \\
&\big((-\tfrac{1097}{2000}, \tfrac{4169}{5000}), (0,1)\big), \quad \big((\tfrac{1527}{2000}, \tfrac{6429}{10000}), (0,2)\big), \quad \big((\tfrac{3831}{5000}, \tfrac{6413}{10000}), (0,2)\big), \\
&\big((\tfrac{3699}{5000}, \tfrac{6711}{10000}), (0,2)\big), \quad \big((\tfrac{7321}{10000}, \tfrac{6767}{10000}), (0,2)\big), \quad \big((\tfrac{7419}{10000}, \tfrac{1339}{2000}), (0,2)\big), \\
&\big((\tfrac{3683}{5000}, \tfrac{3377}{5000}), (0,2)\big), \quad \big((-\tfrac{1087}{2000}, \tfrac{4183}{5000}), (0,1)\big), \quad \big((-\tfrac{1089}{2000}, \tfrac{8387}{10000}), (0,1)\big), \\
&\big((-\tfrac{1089}{2000}, \tfrac{1677}{2000}), (0,1)\big), \quad \big((\tfrac{1}{10}, \tfrac{7}{5\sqrt{2}}), (0,1)\big), \\
&\big((\tfrac{1}{100}(50 + \sqrt{1534}), -\tfrac{-100+\sqrt{1534}}{100\sqrt{2}}), (0,1)\big), \big((\tfrac{9}{10}, \tfrac{3}{5\sqrt{2}}), (0,1)\big)\Big),
\end{aligned}$$

$$\big(((x_{11,1},y_{11,1}),(a_{11,1},b_{11,1})),\dots,((x_{11,47},y_{11,47}),(a_{11,47},b_{11,47})\big) := \Big($$
$$((1,0),(-2,0))\ \big(\big(-\tfrac{529}{4023},\tfrac{22378908}{45415717}\big),(0,1)\big),\quad \big(\big(\tfrac{25699}{75158},\tfrac{11951}{22586}\big),(2,0)\big),$$
$$\big(\big(\tfrac{122233}{192089},\tfrac{5593}{12399}\big),(0,1)\big),\quad \big(\big(\tfrac{6229}{23994},\tfrac{22353}{28738}\big),(0,9)\big),\quad \big(\big(\tfrac{2039}{57213},\tfrac{17365}{20941}\big),(0,1)\big),$$
$$\big(\big(\tfrac{3099}{4183},\tfrac{442047}{1060847}\big),(0,1)\big),\quad \big(\big(-\tfrac{39923}{156499},\tfrac{22371}{26896}\big),(0,1)\big),\quad \big(\big(\tfrac{4038}{5203},\tfrac{4722}{11383}\big),(0,1)\big),$$
$$\big(\big(\tfrac{285}{406},\tfrac{752}{1417}\big),(0,1)\big),\quad \big(\big(\tfrac{15765}{22453},\tfrac{431}{725}\big),(0,1)\big),\quad \big(\big(\tfrac{2023}{7895},\tfrac{2634}{2981}\big),(0,1)\big),$$
$$\big(\big(-\tfrac{810241}{3496246},\tfrac{662044}{743591}\big),(0,1)\big),\quad \big(\big(\tfrac{127129}{185005},\tfrac{42539}{67882}\big),(0,4)\big),\quad \big(\big(-\tfrac{109151}{435226},\tfrac{1106}{1235}\big),$$
$$(0,1)\big),\big(\big(\tfrac{1499}{5037},\tfrac{10953}{12284}\big),(0,1)\big),\big(\big(-\tfrac{8495}{29356},\tfrac{259913}{290617}\big),(0,1)\big),\big(\big(\tfrac{755}{851},\tfrac{3083}{7406}\big),(0,1)\big),$$
$$\big(\big(-\tfrac{15483}{32584},\tfrac{4513239}{5265740}\big),(0,1)\big),\big(\big(-\tfrac{39752315}{80135632},\tfrac{1130}{1337}\big),(0,1)\big),\big(\big(-\tfrac{45318560}{90412991},\tfrac{235960}{280199}\big),$$
$$(0,1)\big),\quad \big(\big(-\tfrac{422566}{838723},\tfrac{6443}{7665}\big),(0,1)\big),\quad \big(\big(-\tfrac{7361}{14390},\tfrac{105082}{125711}\big),(0,1)\big),$$
$$\big(\big(-\tfrac{724614}{1438463},\tfrac{2019}{2369}\big),(0,1)\big),\big(\big(-\tfrac{4861}{9600},\tfrac{1020}{1199}\big),(0,1)\big),\big(\big(-\tfrac{1064}{2059},\tfrac{166081}{196678}\big),(0,1)\big),$$
$$\big(\big(-\tfrac{545}{1034},\tfrac{168253}{200773}\big),(0,1)\big),\ \big(\big(\tfrac{13}{50},\tfrac{24}{25}\big),(0,1)\big),\ \big(\big(\tfrac{13}{51},\tfrac{49}{51}\big),(0,1)\big),\ \big(\big(-\tfrac{45}{82},\tfrac{34}{41}\big),$$
$$(0,1)\big),\ \big(\big(-\tfrac{1135}{2048},\tfrac{1699}{2048}\big),(0,1)\big),\ \big(\big(-\tfrac{1125}{2048},\tfrac{851}{1024}\big),(0,1)\big),\ \big(\big(-\tfrac{1123}{2048},\tfrac{1701}{2048}\big),$$
$$(0,1)\big),\ \big(\big(-\tfrac{1083}{2048},\tfrac{869}{1024}\big),(0,1)\big),\ \big(\big(-\tfrac{1075}{2048},\tfrac{433}{512}\big),(0,1)\big),\ \big(\big(-\tfrac{1069}{2048},\tfrac{873}{1024}\big),$$
$$(0,1)\big),\big(\big(-\tfrac{531}{1024},\tfrac{1745}{2048}\big),(0,1)\big),\big(\big(-\tfrac{529}{1024},\tfrac{875}{1024}\big),(0,1)\big),\big(\big(\tfrac{505}{2048},\tfrac{991}{1024}\big),(0,1)\big),$$
$$\big(\big(\tfrac{511}{2048},\tfrac{1983}{2048}\big),(0,1)\big),\quad \big(\big(\tfrac{513}{2048},\tfrac{991}{1024}\big),(0,1)\big),\quad \big(\big(\tfrac{135}{512},\tfrac{987}{1024}\big),(0,1)\big),$$
$$\big(\big(\tfrac{129106}{516339},\tfrac{2147435}{2219844}\big),(0,1)\big),\quad \big(\big(\tfrac{1}{212}\big(-140+\sqrt{573}\big),\tfrac{\sqrt{11}}{4}\big),$$
$$(0,3)\big),\quad \big(\big(\tfrac{-550-\sqrt{42130}}{1500},\tfrac{\sqrt{11}\big(-25+2\sqrt{42130}\big)}{1500}\big),(0,1)\big),$$
$$\big(\big(\tfrac{1}{48}\big(-33+\sqrt{93}\big),\tfrac{1}{48}\sqrt{11}\big(3+\sqrt{93}\big)\big),(0,1)\big),\quad \big(\big(\tfrac{1639+\sqrt{10021}}{6600},\tfrac{539+\sqrt{10021}}{200\sqrt{11}}\big),$$
$$(0,1)\big)\Big),$$

and let $C_0^{(2)}(k)$ *denote the ultimate period of the orbit of* $(a_{2,k},b_{2,k})_2$ *under* $\mathcal{T}_{2,(x_{2,k},y_{2,k})}$ *for all* $k\in\{1,\dots 45\}$ *and* $C_0^{(11)}(k)$ *the ultimate period of the orbit of* $(a_{11,k},b_{11,k})_{11}$ *under* $\mathcal{T}_{11,(x_{11,k},y_{11,k})}$ *for all* $k\in\{1,\dots 47\}$. *Furthermore let for all* $k\in\mathbb{Z}$*:*

$$\begin{aligned} C_1^{(d)}(k) &:= ((-k,1)_d,(k,-1)_d)\\ C_2^{(d)}(k) &:= ((-k,1)_d,(k+1,-1)_d). \end{aligned}$$

Theorem 5. *The sets* $\mathcal{D}_{1,2}^{(0)}$ *and* $\mathcal{D}_{1,11}^{(0)}$ *do not contain any weakly critical points (and thus no critical points)* r *satisfying* $r\in\overline{\mathcal{D}_{1,2}^{(0)}}$ *and* $r\in\overline{\mathcal{D}_{1,11}^{(0)}}$ *respectively. More precisely the circle of radius* 0.99 *around the origin contains the sets* $\mathcal{D}_{1,2}^{(0)}$ *and* $\mathcal{D}_{1,11}^{(0)}$.

Proof. For any cycle π of complex numbers let $\overline{\pi}$ denote the cycle one gets if all elements of π are replaced by their complex conjugates. The cutout sets of the cycles $C_1^{(2)}(k),C_2^{(2)}(k),\ k\in\mathbb{Z},\ C_0^{(2)}(1),\dots,C_0^{(2)}(45),\ \overline{C_0^{(2)}(1)},\dots,\overline{C_0^{(2)}(45)}$, and $C_1^{(11)}(k),C_2^{(11)}(k),\ k\in\mathbb{Z},\ C_0^{(11)}(1),\dots,C_0^{(11)}(47),\ \overline{C_0^{(11)}(1)},\dots,\overline{C_0^{(11)}(47)}$ respectively, completely cover the ring centered at the origin in the complex plane with inner radius $\frac{99}{100}$ and outer radius 1. Figures 3 and 3 show the cutout sets for the cases $d=2$ and $d=11$ respectively. The list has been found by a combination of a variant of Algorithm 1 with manual search. □

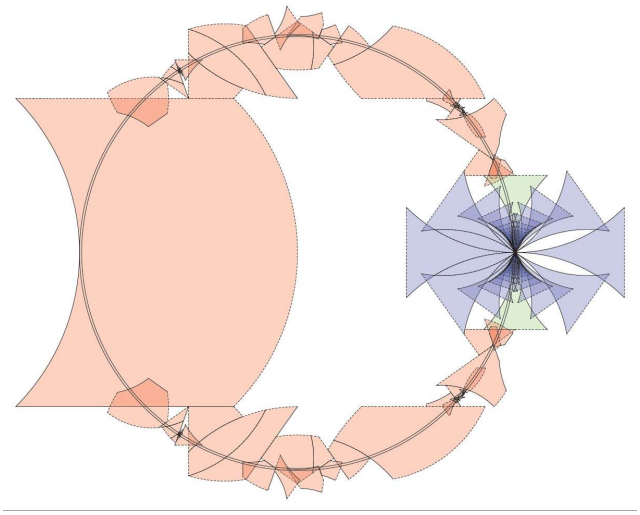

Figure 3: Cutout areas of $\mathcal{D}_{1,2}$ which covers the annulus with radii 99/100 and 1. The green area represents the first cutout area, the blue ones are the two infinite sequences.

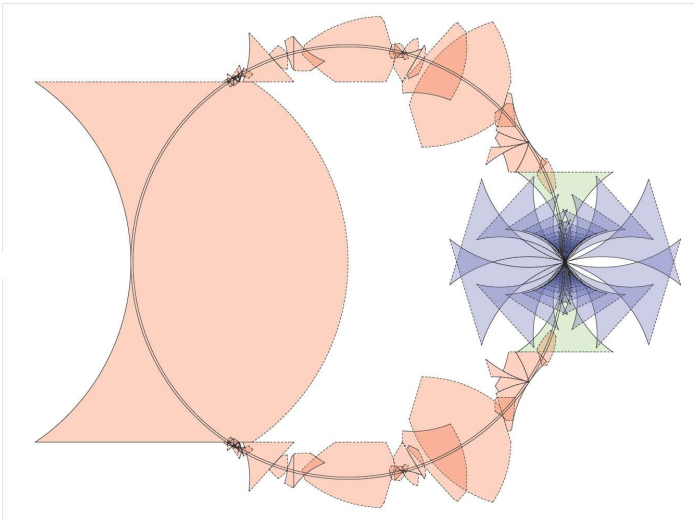

Figure 4: Cutout areas of $\mathcal{D}_{1,11}$ which covers the annulus with radii 99/100 and 1. The green area represents the first cutout area, the blue ones are the two infinite sequences.

4 Conclusion and further work

In this paper shift radix systems have been defined over the complex field (Definition 4), and the one dimensional case has been investigated more precisely.
This can be continued to investigate polynomials and vectors with greater degree, Hausdorff dimensions can be calculated more precisely, or SRS over other Euclidean domains can be investigated as well.

References

[1] S. Akiyama, T. Borbély, H. Brunotte, A. Pethő, and J. M. Thuswaldner. Generalized radix representations and dynamical systems. I. *Acta Math. Hungar.*, 108(3):207–238, 2005.

[2] S. Akiyama, H. Brunotte, A. Pethő, and J. M. Thuswaldner. Generalized radix representations and dynamical systems. II, *Acta Arith.*, 121:21–61, 2006.

[3] H. Brunotte, P. Kirschenhofer and J. M. Thuswaldner. Shift radix systems for Gaussian integers and Pethő's loudspeaker. *Publ. Math. Debrecen*, 79:341–356, 2011.

[4] H. Brunotte. On trinomial bases of radix representations of algebraic integers. *Acta Sci. Math. (Szeged)*, 67:407–413, 2001.

[5] P. Burcsi and A. Kovács. Exhaustive search methods for CNS polynomials. *Monatsh. Math.*, 155(3–4):421–430, 2008.

[6] Hua Loo Keng. *Introduction to number theory.* Springer Verlag Berlin Heidelberg New York, 1982.

[7] M.-A. Jacob and J.-P. Reveilles. *Gaussian numeration systems.* Actes du colloque de Géométrie Discrète DGCI, 1995.

[8] I. Kátai and J. Szabó. Canonical number systems for complex integers. *Acta Sci. Math. (Szeged)*, 37:255–260, 1975.

[9] A. Pethő. On a polynomial transformation and its application to the construction of a public key cryptosystem. *Computational number theory* (Debrecen, 1989), pages 31–43, de Gruyter, Berlin, 1991.

[10] A. Pethő, P. Varga. Canonical number systems over imaginary quadratic Euclidean domains. *Submitted.*

[11] A. Rényi. Representations for real numbers and their ergodic properties. *Acta Math. Acad. Sci. Hungar.*, 8:477-493, 1957.

[12] P. Surer. Characterization results for shift radix systems. Mat. Pannon., 18:265–297, 2007.

[13] M. Weitzer. Characterization algorithms for shift radix systems with finiteness property. *Int. J. Number Theory*, 11:211–232, 2015.

[14] M. Weitzer. On the characterization of Pethő's Loudspeaker. *Publ. Math. Debrecen.*, to appear.

Received 23rd June 2015

Acta Cybernetica **22** (2015) 499–515.

Rectangular Algebras as Tree Recognizers

Magnus Steinby*

To the memory of Ferenc Gécseg

Abstract

We consider finite rectangular algebras of finite type as tree recognizers. The type is represented by a ranked alphabet Σ. We determine the varieties of finite rectangular Σ-algebras and show that they form a Boolean lattice in which the atoms are minimal varieties of finite Σ-algebras consisting of projection algebras. We also describe the corresponding varieties of Σ-tree languages and compare them with some other varieties studied in the literature. Moreover, we establish the solidity properties of these varieties of finite algebras and tree languages. Rectangular algebras have been previously studied by R. Pöschel and M. Reichel (1993), and we make use of some of their results.

1 Introduction

In a projection algebra every fundamental operation is a projection operation. Pöschel and Reichel [11] defined rectangular τ-algebras as the members of the variety generated by all projection algebras of type τ. Rectangular algebras are also natural generalizations of rectangular bands; the rectangular algebras of type $\langle 2 \rangle$ are precisely the rectangular bands.

In this paper we study projection algebras and rectangular algebras as tree recognizers. Hence the algebras considered are finite and of a finite type, represented here by a ranked alphabet Σ. Our general framework is the variety theory of tree languages [12, 13], which establishes bijective correspondences between the varieties Σ-tree languages (Σ-VTLs), the varieties of finite Σ-algebras (Σ-VFAs), and the Σ-varieties of finite congruences (Σ-VFCs).

The class of all finite projection Σ-algebras is not a Σ-VFA, but it contains certain simple Σ-VFAs from which all the Σ-VFAs to be considered here are obtained. Each such atomic Σ-VFA corresponds to some so-called projection alphabet. For any projection alphabet Λ, the class $\mathbf{FProj}_\Lambda$ of all finite Λ-projection algebras is a minimal Σ-VFA, and these Σ-VFAs $\mathbf{FProj}_\Lambda$ are the atoms of the Boolean lattice of all sub-VFAs of the Σ-VFA $\mathbf{FRA}_\Sigma$ of all finite rectangular Σ-algebras. Every sub-VFA $\mathbf{FRA}_\mathcal{L}$ of $\mathbf{FRA}_\Sigma$ corresponds to a set $\mathcal{L}$ of projection alphabets, and it

*Department of Mathematics and Statistics, University of Turku, FIN-20014 Turku, Finland. E-mail: steinby@utu.fi

is the finite join of the Σ-VFAs $\mathbf{FProj}_\Lambda$ such that $\Lambda \in \mathcal{L}$. We also describe the Σ-VFCs that correspond to the Σ-VFAs $\mathbf{FProj}_\Lambda$ and $\mathbf{FRA}_\mathcal{L}$.

It is easy to describe a tree language recognized by a projection algebra; whether a tree is in it depends just on the label of the leaf at the end of a certain path determined by the projection alphabet of the algebra. This observation leads to a simple characterization of the members of the Σ-VTLs $FProj_\Lambda$ that correspond to the Σ-VFAs $\mathbf{FProj}_\Lambda$. Moreover, we show that any tree language in $FProj_\Lambda$ is also recognized by a two-element Λ-projection algebra $\mathcal{P}_\Lambda$, which therefore is (up to isomorphism) the only nontrivial syntactic algebra in $\mathbf{FProj}_\Lambda$. We also note that the syntactic monoid of any member of $FProj_\Lambda$ is either trivial or isomorphic to a certain 3-element monoid. The Σ-VTLs $FRA_\mathcal{L}$ that correspond to the more general Σ-VFAs $\mathbf{FRA}_\mathcal{L}$ are shown to be the ring closures of the unions of the atomic Σ-VTLs $FProj_\Lambda$ they contain. It is also noted that the membership problem is decidable for these Σ-VTLs.

Although the tree languages recognized by rectangular algebras have rather simple descriptions, their trees are not characterized by any local properties. Therefore the Σ-VTLs $FRA_\mathcal{L}$ have little in common with many of the Σ-VTLs previously considered in the literature. Thus we show that the intersection of any $FRA_\mathcal{L}$ with any one of the Σ-VTLs of nilpotent, definite, reverse definite, generalized definite or locally testable Σ-tree languages is just the trivial Σ-VTL. Of course, the corresponding facts hold for Σ-VFAs. On the other hand, we show that FRA_Σ is contained in the Σ-VFA of all aperiodic Σ-tree languages. As another exception, we show that for any projection alphabet Λ, the Σ-VTL $FProj_\Lambda$ is contained in the family $DRec_\Sigma$ of Σ-tree languages recognized by deterministic top-down tree recognizers. This implies that FRA_Σ is contained in the Σ-VTL generated by $DRec_\Sigma$.

We also study the solidity properties of our Σ-VFAs and Σ-VTLs. Graczyńska and Schweigert [7] noted that the solidity of a class of algebras can be defined in terms of derived algebras. A derived algebra $\varkappa(\mathcal{A})$ of a Σ-algebra $\mathcal{A}$ is obtained by replacing each fundamental operation of $\mathcal{A}$ with a term operation determined by the given hypersubstitution $\varkappa$, and a class $\mathbf{K}$ of Σ-algebras is solid if it contains all derived algebras of its members. A family of Σ-tree languages is said to be solid, if it is closed under inverse tree homomorphisms. In fact, we consider the more refined notions of solidity with respect to a given class of hypersubstitutions. In [11] it was shown that the rectangular Σ-algebras form the least nontrivial solid variety of Σ-algebras, and hence it is to be expected that $\mathbf{FRA}_\Sigma$ is the least nontrivial solid Σ-VFA. Also the Σ-VFA of trivial Σ-algebras is naturally solid, but the remaining sub-VFAs of $\mathbf{FRA}_\Sigma$ are shown to have very weak solidity properties. The corresponding facts hold for the Σ-VTLs $FRA_\mathcal{L}$.

2 Preliminaries

We may write $A := B$ to emphasize that A is defined to be B. For any integer $n \geq 0$, let $[n] := \{1, \ldots, n\}$. The set of all subsets of a set A is denoted by $\wp(A)$.

For any relation $\rho \subseteq A \times B$, the fact that $(a, b) \in \rho$ for some $a \in A$ and $b \in B$, will usually be expressed by writing $a \,\rho\, b$. For a mapping $\varphi : A \to B$, we may write the image $\varphi(a)$ of an element $a \in A$ as $a\varphi$. Especially homomorphisms are written this way as right operators that are composed from left to right, i.e., the composition of $\varphi : A \to B$ and $\psi : B \to C$ is written as $\varphi\psi$.

Next we recall some basic matters concerning algebras, tree recognizers and tree languages. For details and further references, cf. [1, 5, 6, 13], for example.

A *ranked alphabet* Σ is a finite set of symbols each of which has a unique positive integer *arity*. For any $m \geq 1$, the set of m-ary symbols in Σ is denoted by Σ_m. Note that we assume that there are no nullary symbols. If $\Sigma = \Sigma_1$, then Σ is said to be *unary*. The *rank type* of Σ is the set $r(\Sigma) := \{m \mid \Sigma_m \neq \emptyset\}$. The ranked alphabet Σ will have two roles. Firstly, the inner nodes of trees are labeled with symbols from Σ. Secondly, Σ is a finite set of operation symbols that determines the type of the algebras to be considered. To avoid exceptions for the unary case, we make the following general assumption.

Convention. From now on, Σ is a ranked alphabet without nullary symbols that contains at least one symbol of arity ≥ 2.

We also use ordinary finite nonempty alphabets $X, Y, \ldots$ that we call *leaf alphabets*. These are assumed to be disjoint from Σ. For any leaf alphabet X, the set $T_\Sigma(X)$ of Σ-*terms over* X is the smallest set T such that $X \subseteq T$, and $f(t_1, \ldots, t_m) \in T$ whenever $m \in r(\Sigma)$, $f \in \Sigma_m$ and $t_1, \ldots, t_m \in T$. Such terms are regarded in the usual way as labeled trees, and we call them ΣX-*trees*. Subsets of $T_\Sigma(X)$ are called ΣX-*tree languages*. We may also speak about Σ-*trees* and Σ-*tree languages* without specifying the leaf alphabet, or just about *trees* and *tree languages*. A *family of* Σ-*tree languages* is a mapping $\mathcal{V}$ that assigns to every leaf alphabet X a set $\mathcal{V}(X)$ of ΣX-tree languages. We write such a family as $\mathcal{V} = \{\mathcal{V}(X)\}_X$. For any two such families $\mathcal{U}$ and $\mathcal{V}$, we set $\mathcal{U} \subseteq \mathcal{V}$ iff $\mathcal{U}(X) \subseteq \mathcal{V}(X)$ for every X. Unions and intersections of families of Σ-tree languages are defined by similar componentwise conditions.

Let ξ be a special symbol that does not appear in Σ or X. A $\Sigma(X \cup \{\xi\})$-tree in which ξ appears exactly once, is called a ΣX-*context*. The set of all ΣX-contexts is denoted by $C_\Sigma(X)$. If $p, q \in C_\Sigma(X)$ and $t \in T_\Sigma(X)$, then $p \cdot q = q(p)$ and $t \cdot q = q(t)$ are the ΣX-context and the ΣX-tree obtained from q by replacing the ξ in it with p or t, respectively. Clearly, $C_\Sigma(X)$ forms a monoid for the product $p \cdot q$ and the identity element ξ.

A Σ-*algebra* $\mathcal{A}$ consists of a nonempty set A and a Σ-indexed family $(f^{\mathcal{A}} \mid f \in \Sigma)$ such that if $f \in \Sigma_m$, then $f^{\mathcal{A}} : A^m \to A$ is an m-ary operation on A. We write simply $\mathcal{A} = (A, \Sigma)$. Subalgebras, homomorphisms, (epimorphic) images and direct products are defined as usual. An algebra $\mathcal{B}$ is said to *cover* an algebra $\mathcal{A}$ if $\mathcal{A}$ is an image of a subalgebra of $\mathcal{B}$. This we express by writing $\mathcal{A} \preceq \mathcal{B}$. The ΣX-trees form the ΣX-*term algebra* $\mathcal{T}_\Sigma(X) = (T_\Sigma(X), \Sigma)$, where $f^{\mathcal{T}_\Sigma(X)}(t_1, \ldots, t_m) = f(t_1, \ldots, t_m)$ for all $m \in r(\Sigma)$, $f \in \Sigma_m$ and $t_1, \ldots, t_m \in T_\Sigma(X)$.

A (*deterministic bottom-up*) ΣX-*recognizer* $\mathbf{A} = (\mathcal{A}, \alpha, F)$ consists of a finite Σ-algebra $\mathcal{A} = (A, \Sigma)$, the elements of which are called *states*, an *initial assignment*

$\alpha : X \to A$ that specifies the starting states at the leaves, and a set $F \subseteq A$ of *final states*. The root of a ΣX-tree t is reached in state $t\alpha_{\mathcal{A}}$, where $\alpha_{\mathcal{A}} : \mathcal{T}_\Sigma(X) \to \mathcal{A}$ is the homomorphic extension of α, and hence the ΣX-tree language *recognized* by $\mathbf{A}$ is defined as $T(\mathbf{A}) = \{t \in T_\Sigma(X) \mid t\alpha_{\mathcal{A}} \in F\}$.

A ΣX-tree language is called *recognizable*, or *regular*, if it is recognized by a ΣX-recognizer. Let $Rec_\Sigma(X)$ be the set of all recognizable ΣX-tree languages, and let $Rec_\Sigma = \{Rec_\Sigma(X)\}_X$ be the family of recognizable Σ-tree languages. We may also say that a Σ-algebra $\mathcal{A} = (A, \Sigma)$ *recognizes* a ΣX-tree language T if $T = F\varphi^{-1}$ for some homomorphism $\varphi : \mathcal{T}_\Sigma(X) \to \mathcal{A}$ and some $F \subseteq A$. Obviously, a ΣX-tree language is recognized by a finite algebra iff it is regular.

The following review of the variety theory of tree languages follows [12] and [13], where also further references can be found. The *syntactic algebra* of a ΣX-tree language T is the quotient algebra $\mathrm{SA}(T) := \mathcal{T}_\Sigma(X)/\theta_T$, where θ_T is the *syntactic congruence* of T defined by

$$s\,\theta_T\,t \;\Leftrightarrow\; (\forall p \in C_\Sigma(X))(p(s) \in T \;\leftrightarrow\; p(t) \in T) \quad (s, t \in T_\Sigma(X)).$$

It is easy to see that $\mathrm{SA}(T)$ is the minimal Σ-algebra recognizing T in the sense that a Σ-algebra $\mathcal{A}$ recognizes T iff $\mathrm{SA}(T) \preceq \mathcal{A}$.

A *variety of Σ-tree languages* (Σ-VTL) is a family of Σ-tree languages $\mathcal{V} = \{\mathcal{V}(X)\}_X$ such that for all leaf alphabets X and Y,

(V1) $\mathcal{V}(X)$ is a Boolean subalgebra of $Rec_\Sigma(X)$,

(V2) if $T \in \mathcal{V}(X)$ and $p \in C_\Sigma(X)$, then $p^{-1}(T) := \{t \in T_\Sigma(X) \mid p(t) \in T\} \in \mathcal{V}(X)$, and

(V3) if $T \in \mathcal{V}(Y)$, then $T\varphi^{-1} := \{t \in T_\Sigma(X) \mid t\varphi \in T\}$ is in $\mathcal{V}(X)$ for every homomorphism $\varphi : \mathcal{T}_\Sigma(X) \to \mathcal{T}_\Sigma(Y)$.

The least Σ-VTL is $Triv_\Sigma = \{Triv_\Sigma(X)\}_X$, where $Triv_\Sigma(X) = \{\emptyset, T_\Sigma(X)\}$, and the greatest Σ-VTL is $Rec_\Sigma = \{Rec_\Sigma(X)\}_X$.

A class of finite Σ-algebras $\mathbf{K}$ is called a *variety of finite Σ-algebras* (Σ-VFA) (or a *pseudovariety*) if it is closed under subalgebras, epimorphic images and finite direct products, i.e., if $S(\mathbf{K}), H(\mathbf{K}), P_f(\mathbf{K}) \subseteq \mathbf{K}$. The *$\Sigma$-VFA generated* by a class $\mathbf{K}$ of finite Σ-algebras is denoted by $V_f(\mathbf{K})$. Since $V_f(\mathbf{K}) = HSP_f(\mathbf{K})$, a Σ-algebra $\mathcal{A}$ is in $V_f(\mathbf{K})$ iff $\mathcal{A} \preceq \mathcal{A}_1 \times \ldots \times \mathcal{A}_n$ for some $n \geq 0$ and algebras $\mathcal{A}_1, \ldots, \mathcal{A}_n \in \mathbf{K}$. Let $\mathbf{Triv}_\Sigma$ be the Σ-VFA of all trivial Σ-algebras.

For any Σ and X, let $\mathrm{FC}_\Sigma(X) := \{\theta \in \mathrm{Con}(\mathcal{T}_\Sigma(X)) \mid T_\Sigma(X)/\theta \,\text{finite}\}$ be the set of *finite congruences* of $\mathcal{T}_\Sigma(X)$. If Γ assigns to each leaf alphabet a subset $\Gamma(X)$ of $\mathrm{FC}_\Sigma(X)$, we write $\Gamma = \{\Gamma(X)\}_X$, and we call Γ a *Σ-variety of finite congruences* (Σ-VFC) if for all X and Y,

(C1) $\Gamma(X)$ is a filter of the lattice $(\mathrm{FC}_\Sigma(X), \subseteq)$, and

(C2) $\varphi \circ \theta \circ \varphi^{-1} := \{(s, t) \mid s, t \in T_\Sigma(X), s\varphi\,\theta\,t\varphi\}$ belongs to $\Gamma(X)$ for every $\theta \in \Gamma(Y)$ and every homomorphism $\varphi : \mathcal{T}_\Sigma(X) \to \mathcal{T}_\Sigma(Y)$.

The three classes of varieties defined above form complete lattices with respect to the natural inclusion relations. They are connected by three pairs of mutually inverse isomorphisms.

For any Σ-VFA $\mathbf{K}$, let $\mathbf{K}^t$ be the family of Σ-tree languages and let $\mathbf{K}^c$ be the Σ-family of finite congruences such that for each X, $\mathbf{K}^t(X) = \{T \subseteq T_\Sigma(X) \mid \mathrm{SA}(T) \in \mathbf{K}\}$ and $\mathbf{K}^c(X) = \{\theta \in \mathrm{FC}_\Sigma(X) \mid \mathcal{T}_\Sigma(X)/\theta \in \mathbf{K}\}$. For any Σ-VTL $\mathcal{V} = \{\mathcal{V}(X)\}_X$, let $\mathcal{V}^a$ be the Σ-VFA generated by the syntactic algebras of the tree languages belonging to $\mathcal{V}$, and let $\mathcal{V}^c$ be the Σ-family of finite congruences such that for any X, $\mathcal{V}^c(X) := [\{\theta_T \mid T \in \mathcal{V}(\Sigma, X)\})$ is the filter of $\mathrm{FC}_\Sigma(X)$ generated by the syntactic congruences of the members of $\mathcal{V}(X)$. Finally, for any Σ-VFC $\Gamma = \{\Gamma(X)\}_X$, let $\Gamma^a := V_f(\{\mathcal{T}_\Sigma(X)/\theta \mid \theta \in \Gamma(X) \text{ for some } X\})$ and let Γ^t be the family of Σ-tree languages such that for any X, $\Gamma^t(X) = \{T \subseteq T_\Sigma(X) \mid \theta_T \in \Gamma(X)\}$. The Variety Theorem for Σ-tree languages can now be stated as follows.

Theorem 2.1. *The mappings $\mathbf{K} \mapsto \mathbf{K}^t$, $\mathcal{V} \mapsto \mathcal{V}^a$, $\mathbf{K} \mapsto \mathbf{K}^c$, $\Gamma \mapsto \Gamma^a$, $\mathcal{V} \mapsto \mathcal{V}^c$, and $\Gamma \mapsto \Gamma^t$ form three pairs of mutually inverse isomorphisms between the lattices of all Σ-VFAs, Σ-VTLs and Σ-VFCs.*

A Σ-VFC $\Gamma = \{\Gamma(X)\}_X$ is *principal* if for every X, $\Gamma(X)$ is a principal filter in $\mathrm{FC}_\Sigma(X)$. It is easy to see that a family $\Gamma = \{[\gamma_X)\}_X$, where $\gamma_X \in \mathrm{FC}_\Sigma(X)$ for each X, is a principal Σ-VFC iff for all X and Y, $\gamma_X \subseteq \varphi \circ \gamma_Y \circ \varphi^{-1}$ for every homomorphism $\varphi : \mathcal{T}_\Sigma(X) \to \mathcal{T}_\Sigma(Y)$.

Remark 2.1. If $\Gamma = \{[\gamma_X)\}_X$ is a principal Σ-VFC, then $\Gamma^t(X)$ is the finite set of ΣX-tree languages saturated by γ_X. Conversely, if $\mathcal{V} = \{\mathcal{V}(X)\}_X$ is a Σ-VTL such that $\mathcal{V}(X)$ is a finite set for every X, then $\mathcal{V}^c$ is a principal Σ-VFC because the filter $\mathcal{V}^c(X)$ is generated by the syntactic congruences of the members of $\mathcal{V}(X)$.

The join of any finite set of Σ-VFAs can be described as follows.

Lemma 2.1. *For any Σ-VFAs $\mathbf{K}_1, \ldots, \mathbf{K}_n$ $(n \geq 1)$, the join $\mathbf{K}_1 \vee \ldots \vee \mathbf{K}_n = V_f(\mathbf{K}_1 \cup \ldots \cup \mathbf{K}_n)$ consists of all Σ-algebras $\mathcal{A}$ such that $\mathcal{A} \preceq \mathcal{A}_1 \times \ldots \times \mathcal{A}_n$ for some $\mathcal{A}_1 \in \mathbf{K}_1, \ldots, \mathcal{A}_n \in \mathbf{K}_n$.*

The *Σ-VTL generated* by a family of recognizable Σ-tree languages $\mathcal{V}$ is the least Σ-VTL containing $\mathcal{V}$. The *Boolean closure* $\mathfrak{B}\mathcal{V}$ and the *ring closure* $\mathfrak{R}\mathcal{V}$ of $\mathcal{V}$ are the families of Σ-tree languages such that for any X, $\mathfrak{B}\mathcal{V}(X)$ is the Boolean closure of $\mathcal{V}(X)$ in $Rec_\Sigma(X)$ and $\mathfrak{R}\mathcal{V}(X)$ is the least subset of $Rec_\Sigma(X)$ containing $\mathcal{V}(X)$ and closed under finite intersections and unions.

Lemma 2.2. *If a family of recognizable Σ-tree languages $\mathcal{V} = \{\mathcal{V}(X)\}_X$ satisfies conditions (V2) and (V3), then $\mathfrak{B}\mathcal{V}$ is the Σ-VTL generated by $\mathcal{V}$. If, moreover, $T \in \mathcal{V}(X)$ implies $T^{\complement} \in \mathcal{V}(X)$ for every X, then $\mathfrak{R}\mathcal{V}$ is the Σ-VTL generated by $\mathcal{V}$.*

Proof. The lemma follows from the identities $p^{-1}(T \cup T') = p^{-1}(T) \cup p^{-1}(T')$, $p^{-1}(T^{\complement}) = p^{-1}(T)^{\complement}$, $(T \cup T')\varphi^{-1} = T\varphi^{-1} \cup T'\varphi^{-1}$ and $T^{\complement}\varphi^{-1} = (T\varphi^{-1})^{\complement}$, where p and φ are as in (V2) and (V3) and T and T' are tree languages of the appropriate kind. □

The join $\mathcal{V}_1 \vee \ldots \vee \mathcal{V}_n$ of any Σ-VTLs $\mathcal{V}_1 = \{\mathcal{V}_1(X)\}_X, \ldots, \mathcal{V}_n = \{\mathcal{V}_n(X)\}_X$ $(n \geq 1)$ is naturally the Σ-VTL generated by the union $\mathcal{V}_1 \cup \ldots \cup \mathcal{V}_n = \{\mathcal{V}_1(X) \cup \ldots \cup \mathcal{V}_n(X)\}_X$. Since the $\mathcal{V}_i$s are Σ-VTLs, this union satisfies the conditions of Lemma 2.2, and we get

Corollary 2.1. *If $\mathcal{V}_1, \ldots, \mathcal{V}_n$ $(n \geq 1)$ are Σ-VTLs, then $\mathcal{V}_1 \vee \ldots \vee \mathcal{V}_n = \mathfrak{R}(\mathcal{V}_1 \cup \ldots \cup \mathcal{V}_n)$.*

3 Projection algebras and rectangular algebras

For any $m > 0$ and $i \in [m]$, the i^{th} *m-ary projection operation* on a set A is the mapping $e_i^m : A^m \to A, (a_1, \ldots, a_m) \mapsto a_i$. (We omit A from the notation as it is always known from the context.) An algebra $\mathcal{A} = (A, \Sigma)$ is called [11] a *projection algebra* if for all $m \in r(\Sigma)$ and $f \in \Sigma_m$, there is an $i \in [m]$ for which $f^{\mathcal{A}} = e_i^m$. Let $\mathbf{FProj}_\Sigma$ denote the class of all finite projection Σ-algebras. The direct product of projection algebras is in general not a projection algebra, but we shall show that $\mathbf{FProj}_\Sigma$ contains subclasses that are Σ-VFAs.

The *path alphabet* of Σ is the set $\widehat{\Sigma} := \bigcup\{\Sigma_m \times [m] \mid m \in r(\Sigma)\}$ regarded as an ordinary alphabet. We shall write f_i for (f, i). Words over $\widehat{\Sigma}$ describe paths in trees; if f_i appears in such a word, then f labels a node on the path and i indicates the direction taken at that node. We call a subalphabet Λ of $\widehat{\Sigma}$ a *projection alphabet* if for all $m \in r(\Sigma)$ and $f \in \Sigma_m$, there is exactly one $i \in [m]$ such that $f_i \in \Lambda$. Let $pa(\Sigma)$ denote the set of all projection alphabets over Σ. If $\Lambda \in pa(\Sigma)$, the *Λ-path* $\Lambda(t)$ in a ΣX-tree t is defined as follows:

(1) $\Lambda(x) = x$ for every $x \in X$;

(2) $\Lambda(t) = f_i\Lambda(t_i)$ if $t = f(t_1, \ldots, t_m)$ and $f_i \in \Lambda$.

Obviously, $\Lambda(t)$ is always of the form wx, where $w \in \Lambda^*$ and $x \in X$. The word w describes a path from the root to a leaf and x is the label of that leaf. Let $\Lambda^\bullet(t)$ denote this label x. Each projection algebra $\mathcal{A}$ defines a projection alphabet

$$\Lambda_{\mathcal{A}} := \{f_i \mid f \in \Sigma_m, m \in r(\Sigma), i \in [m], f^{\mathcal{A}} = e_i^m\},$$

and conversely, given the set A, this projection alphabet determines the projection algebra $\mathcal{A}$.

Definition 3.1. For any $\Lambda \in pa(\Sigma)$, we call a projection algebra $\mathcal{A} = (A, \Sigma)$ a *Λ-projection algebra* if $\Lambda_{\mathcal{A}} = \Lambda$. The class of all finite Λ-projection algebras is denoted by $\mathbf{FProj}_\Lambda$. □

Let $\mathcal{A}$ and $\mathcal{B}$ be projection algebras. It is clear that if $\mathcal{A}$ is a subalgebra or an image of $\mathcal{B}$, then $\Lambda_{\mathcal{A}} = \Lambda_{\mathcal{B}}$. Moreover, it is easy to see that for any projection alphabet $\Lambda \in pa(\Sigma)$ the direct product of any family of Λ-projection algebras is a Λ-projection algebra. Hence, any $\mathbf{FProj}_\Lambda$ is a Σ-VFA. However, we can say a bit more about these classes.

For each $\Lambda \in pa(\Sigma)$, let $\mathcal{P}_\Lambda = (\{0,1\}, \Sigma)$ be the two-element Λ-projection algebra. In [11] it was shown that the two-element projection algebras are the only nontrivial subdirectly irreducible projection algebras. Hence the following proposition is obvious.

Proposition 3.1. *For any projection alphabet $\Lambda \in pa(\Sigma)$, $\mathcal{P}_\Lambda$ is the only nontrivial subdirectly irreducible algebra in $\mathbf{FProj}_\Lambda$, and hence every algebra in $\mathbf{FProj}_\Lambda$ is a finite subdirect power of $\mathcal{P}_\Lambda$. Moreover, $\mathbf{FProj}_\Lambda$ is a minimal Σ-VFA.*

For any $\Lambda \in pa(\Sigma)$ and any leaf alphabet X, let

$$\rho_\Lambda(X) := \{(s,t) \mid s,t \in T_\Sigma(X), \Lambda^\bullet(s) = \Lambda^\bullet(t)\}.$$

It is easy to see that $\rho_\Lambda(X)$ is a congruence on $\mathcal{T}_\Sigma(X)$ and that the $\rho_\Lambda(X)$-classes are precisely the sets $[x] := \{t \in T_\Sigma(X) \mid \Lambda^\bullet(t) = x\}$, where $x \in X$. Hence the quotient algebra $\mathcal{F}_\Lambda(X) := \mathcal{T}_\Sigma(X)/\rho_\Lambda(X)$ has $|X|$ elements. Moreover, for any $m \in r(\Sigma)$, $f \in \Sigma_m$ and $x_1, \ldots, x_m \in X$,

$$f^{\mathcal{F}_\Lambda(X)}([x_1], \ldots, [x_m]) = [x_i],$$

for the $i \in [m]$ such that $f_i \in \Lambda$. Furthermore, it is clear that if $\mathcal{A} = (A, \Sigma)$ is a Λ-projection algebra, then any mapping $\varphi : \{[x] \mid x \in X\} \to A$ is a homomorphism from $\mathcal{F}_\Lambda(X)$ to $\mathcal{A}$. Hence, $\mathcal{F}_\Lambda(X)$ is freely generated over the class of all Λ-projection algebras by the set $\{[x] \mid x \in X\}$. Since $\mathcal{F}_\Lambda(X)$ is finite, it belongs to $\mathbf{FProj}_\Lambda$, and therefore $\rho_\Lambda(X)$ is the least congruence θ on $\mathcal{T}_\Sigma(X)$ such that $\mathcal{T}_\Sigma(X)/\theta \in \mathbf{FProj}_\Lambda$. This means that $\mathbf{FProj}^c_\Lambda$ is the principal Σ-VFC $\{[\rho_\Lambda(X))\}_X$. These observations are summarized by the following proposition.

Proposition 3.2. *For any $\Lambda \in pa(\Sigma)$ and any X, the set $\{[x] \mid x \in X\}$ generates $\mathcal{F}_\Lambda(X)$ freely over $\mathbf{FProj}_\Lambda$. Moreover, $\mathbf{FProj}^c_\Lambda = \{[\rho_\Lambda(X))\}_X$.*

The variety generated by all projection algebras of a given, not necessarily finite, type was studied by Pöschel and Reichel [11] who called its members *rectangular algebras*. Let us note that the rectangular algebras appear also in Ésik [3] in the form of "diagonal theories". We denote by $\mathbf{RA}_\Sigma$ the variety of rectangular Σ-algebras and by $\mathbf{FRA}_\Sigma$ the Σ-VFA formed by the finite rectangular Σ-algebras. Let us now exhibit all the sub-VFAs of $\mathbf{FRA}_\Sigma$.

For any set $\mathcal{L} \subseteq \wp(\widehat{\Sigma})$ of projection alphabets, let $\mathbf{FRA}_\mathcal{L}$ denote the join $V_f(\bigcup_{\Lambda\in\mathcal{L}} \mathbf{FProj}_\Lambda)$ of the Σ-VFAs $\mathbf{FProj}_\Lambda$ with $\Lambda \in \mathcal{L}$. Of course, $\mathbf{FRA}_{pa(\Sigma)} = \mathbf{FRA}_\Sigma$ and $\mathbf{FRA}_{\{\Lambda\}} = \mathbf{FProj}_\Lambda$ for each $\Lambda \in pa(\Sigma)$. The nontrivial subdirectly irreducible members of $\mathbf{FRA}_\mathcal{L}$ are the algebras $\mathcal{P}_\Lambda = (\{0,1\}, \Sigma)$ with $\Lambda \in \mathcal{L}$, and hence $\{\mathcal{P}_\Lambda \mid \Lambda \in \mathcal{L}\}$ is a minimal generating set of $\mathbf{FRA}_\mathcal{L}$. It is also clear that for any $\mathcal{L}, \mathcal{M} \subseteq pa(\Sigma)$,

(1) $\mathbf{FRA}_\mathcal{L} \subset \mathbf{FRA}_\mathcal{M}$ iff $\mathcal{L} \subset \mathcal{M}$, and

(2) $\mathbf{FRA}_\mathcal{L} \cap \mathbf{FRA}_\mathcal{M} = \mathbf{Triv}_\Sigma$ if $\mathcal{L} \cap \mathcal{M} = \emptyset$.

Let $n(\Sigma)$ denote the product of the arities of the symbols in Σ. It is clear that $n(\Sigma)$ is the number of projection alphabets $\Lambda \in pa(\Sigma)$, and therefore also the number the algebras $\mathcal{P}_\Lambda$. As noted in [11], this means that $\mathbf{RA}_\Sigma$ has precisely $2^{n(\Sigma)}$ subvarieties. Using the above observations, we can formulate the corresponding statement for $\mathbf{FRA}_\Sigma$ in the following more detailed form.

Proposition 3.3. *The Σ-VFAs $\mathbf{FRA}_\mathcal{L}$ ($\mathcal{L} \subseteq pa(\Sigma)$) form a $2^{n(\Sigma)}$-element Boolean sublattice of the lattice of all Σ-VFAs. In this sublattice*

(1) *the least element is $\mathbf{Triv}_\Sigma$, the greatest element is $\mathbf{FRA}_\Sigma$,*

(2) $\mathbf{FRA}_\mathcal{L} \vee \mathbf{FRA}_\mathcal{M} = \mathbf{FRA}_{\mathcal{L}\cup\mathcal{M}}$, $\mathbf{FRA}_\mathcal{L} \wedge \mathbf{FRA}_\mathcal{M} = \mathbf{FRA}_{\mathcal{L}\cap\mathcal{M}}$, *and* $\mathbf{FRA}_\mathcal{L}^{\complement} = \mathbf{FRA}_{pa(\Sigma)\setminus\mathcal{L}}$, *for all $\mathcal{L}, \mathcal{M} \subseteq pa(\Sigma)$, and*

(3) *the atoms are the minimal Σ-VFAs $\mathbf{FProj}_\Lambda$ with $\Lambda \in pa(\Sigma)$.*

The join $\mathbf{FRA}^c_\mathcal{L}$ of the Σ-VFCs $\mathbf{FProj}^c_\Lambda$ with $\Lambda \in \mathcal{L}$, is the principal Σ-VFC $\{[\rho_\mathcal{L}(X))\}_X$, where $\rho_\mathcal{L}(X) := \bigcap_{\Lambda\in\mathcal{L}} \rho_\Lambda(X)$ for each X. Hence, the counterpart of Proposition 3.3 for Σ-VFCs can be written as follows.

Corollary 3.1. *The Σ-VFCs $\mathbf{FRA}^c_\mathcal{L}$ form a $2^{n(\Sigma)}$-element Boolean sublattice in the lattice of all Σ-VFCs. In this sublattice the least element is $\{\{\nabla_{T_\Sigma(X)}\}\}_X$, the greatest element is $\{[\rho_{pa(\Sigma)}(X))\}_X$, and for all $\mathcal{L}, \mathcal{M} \subseteq pa(\Sigma)$, $\mathbf{FRA}^c_\mathcal{L} \vee \mathbf{FRA}^c_\mathcal{M} = \{[\rho_{\mathcal{L}\cup\mathcal{M}}(X))\}_X$, $\mathbf{FRA}^c_\mathcal{L} \wedge \mathbf{FRA}^c_\mathcal{M} = \{[\rho_{\mathcal{L}\cap\mathcal{M}}(X))\}_X$, and $(\mathbf{FRA}^c_\mathcal{L})^{\complement} = \{[\rho_{pa(\Sigma)\setminus\mathcal{L}}(X))\}_X$. The atoms of the sublattice are the Σ-VFCs $\mathbf{FProj}^c_\Lambda = \{[\rho_\Lambda(X))\}_X$ with $\Lambda \in pa(\Sigma)$.*

In [11] it was shown that any rectangular algebra of finite type is isomorphic to the direct product of a finite family of projection algebras. In particular, any member of $\mathbf{FRA}_\Sigma$ is isomorphic to the direct product of a finite family of finite projection algebras. By collecting together factors belonging to the same Σ-VFA $\mathbf{FProj}_\Lambda$, this decomposition result can be expressed more precisely as follows.

Proposition 3.4. *For any set of projection alphabets $\mathcal{L} = \{\Lambda_1, \ldots, \Lambda_k\} \subseteq pa(\Sigma)$, every algebra in $\mathbf{FRA}_\mathcal{L}$ is isomorphic to a direct product $\mathcal{A}_1 \times \cdots \times \mathcal{A}_k$ where $\mathcal{A}_i \in \mathbf{FProj}_{\Lambda_i}$ for $i = 1, \ldots, k$.*

4 Projection and rectangular algebras as tree recognizers

We shall now consider the tree languages recognizable by projection algebras and rectangular algebras. For any $\Lambda \in pa(\Sigma)$ and any $\mathcal{L} \subseteq pa(\Sigma)$, let $FProj_\Lambda = \{FProj_\Lambda(X)\}_X$ and $FRA_\mathcal{L} = \{FRA_\mathcal{L}(X)\}_X$ be the Σ-VTLs that correspond to $\mathbf{FProj}_\Lambda$ and $\mathbf{FRA}_\mathcal{L}$, respectively. The Σ-VTL $FRA_{pa(\Sigma)}$ may be denoted also by FRA_Σ.

It is easy to see that any ΣX-tree t can be evaluated in a Λ-projection algebra $\mathcal{A} = (A, \Sigma)$ for an assignment $\alpha : X \to A$ simply by transporting the value $\alpha(\Lambda^\bullet(t))$

along the path described by $\Lambda(t)$ from the leaf to the root. This is expressed formally by the following lemma.

Lemma 4.1. *Let $\mathcal{A} = (A, \Sigma)$ be a Λ-projection algebra and let $\alpha : X \to A$ be any assignment. Then $t\alpha_{\mathcal{A}} = \alpha(\Lambda^\bullet(t))$ for every ΣX-tree t.*

Corollary 4.1. *$T(\mathbf{A}) = \{t \in T_\Sigma(X) \mid \alpha(\Lambda^\bullet(t)) \in F\}$ for any ΣX-recognizer $\mathbf{A} = (\mathcal{A}, \alpha, F)$ such that $\mathcal{A} \in \mathbf{FProj}_\Lambda$.*

For any $Y \subseteq X$, let $T_\Lambda(X, Y) := \{t \in T_\Sigma(X) \mid \Lambda^\bullet(t) \in Y\}$. By Proposition 3.2, the principal Σ-VFC $\{[\rho_\Lambda(X))\}_X$ corresponds to the Σ-VFA $\mathbf{FProj}_\Lambda$, and hence also to the Σ-VTL $FProj_\Lambda$. Clearly, the ΣX-tree languages saturated by $\rho_\Lambda(X)$ are exactly the sets $T_\Lambda(X, Y)$. By Remark 2.1 this fact yields the first part of the following proposition but we shall give a direct proof.

Proposition 4.1. *For any projection alphabet $\Lambda \in pa(\Sigma)$ and any leaf alphabet X,*

$$FProj_\Lambda(X) = \{T_\Lambda(X, Y) \mid Y \subseteq X\}.$$

Moreover, a ΣX-tree language T is in $FProj_\Lambda(X)$ if and only if $\mathrm{SA}(T)$ is either trivial or isomorphic to $\mathcal{P}_\Lambda$.

Proof. Firstly, $T(\mathbf{A})$ in Corollary 4.1 equals $T_\Lambda(X, Y)$ for $Y = \{x \in X \mid \alpha(x) \in F\}$. Conversely, for any given $T_\Lambda(X, Y)$, let $\mathbf{A} = (\mathcal{P}_\Lambda, \alpha, F)$ be the ΣX-recognizer where $\alpha(x) = 1$ for $x \in Y$, $\alpha(x) = 0$ for $x \in X \setminus Y$, and $F = \{1\}$. Then $T(\mathbf{A}) = \{t \in T_\Sigma(X) \mid \alpha(\Lambda^\bullet(t)) = 1\} = \{t \in T_\Sigma(X) \mid \Lambda^\bullet(t) \in Y\} = T_\Lambda(X, Y)$ by Corollary 4.1 and Lemma 4.1.

Let us now consider any tree language $T \subseteq T_\Sigma(X)$. If $\mathrm{SA}(T)$ is trivial or isomorphic to $\mathcal{P}_\Lambda$, then $T \in FProj_\Lambda(X)$ as both the trivial Σ-algebras and $\mathcal{P}_\Lambda$ are in $\mathbf{FProj}_\Lambda$. On the other hand, if $T \in FProj_\Lambda(X)$, then T is recognized by a Λ-projection algebra. By what we have shown above, this means that T is of the form $T_\Lambda(X, Y)$ and therefore recognized by $\mathcal{P}_\Lambda$. If $\mathcal{P}_\Lambda$ is the minimal Σ-algebra recognizing T, then $\mathrm{SA}(T) \cong \mathcal{P}_\Lambda$, but otherwise $\mathrm{SA}(T)$ is trivial. □

Every subdirectly irreducible algebra is syntactic, but the converse does not hold in general. However, Proposition 4.1 shows that here the two classes coincide.

Corollary 4.2. *A projection algebra is syntactic if and only if it is subdirectly irreducible, i.e., iff it is trivial or isomorphic to one of the algebras $\mathcal{P}_\Lambda$ ($\Lambda \in pa(\Sigma)$).*

Although we know by the Variety Theorem 2.1 that $FProj_\Lambda$ is a Σ-VTL, it may be instructive to show also directly that, for any given projection alphabet $\Lambda \in pa(\Sigma)$, the sets $T_\Lambda(X, Y)$ form a Σ-VTL. Firstly, for any X and $Y, Y' \subseteq X$, we have $T_\Lambda(X, Y)^{\complement} = T_\Lambda(X, X \setminus Y)$ and $T_\Lambda(X, Y) \cup T_\Lambda(X, Y') = T_\Lambda(X, Y \cup Y')$. Let us extend the function $\Lambda^\bullet$ to ΣX-contexts in the obvious way. Then we have for any $p \in C_\Sigma(X)$,

$$p^{-1}(T_\Lambda(X, Y)) = \begin{cases} T_\Sigma(X) = T_\Lambda(X, X) & \text{if } \Lambda^\bullet(p) \in Y; \\ \emptyset = T_\Lambda(X, \emptyset) & \text{if } \Lambda^\bullet(p) \in X \setminus Y; \\ T_\Lambda(X, Y) & \text{if } \Lambda^\bullet(p) = \xi. \end{cases}$$

Finally, if $\varphi : \mathcal{T}_\Sigma(X) \to \mathcal{T}_\Sigma(Y)$ is a homomorphism and $Y' \subseteq Y$, then $T_\Lambda(Y, Y')\varphi^{-1} = T_\Lambda(X, X')$, where $X' = \{x \in X \mid \Lambda^\bullet(x\varphi) \in Y'\}$.

Following [15], we define the *syntactic monoid congruence* of a ΣX-tree language T as the relation μ_T on $C_\Sigma(X)$ such that

$$p\,\mu_T\,q \;\Leftrightarrow\; (\forall t \in T_\Sigma(X))(\forall r \in C_\Sigma(X))(t \cdot p \cdot r \in T \;\leftrightarrow\; t \cdot q \cdot r \in T) \quad (p, q \in C_\Sigma(X)),$$

and the *syntactic monoid* $\mathrm{SM}(T)$ of T as the quotient monoid $C_\Sigma(X)/\mu_T$.

It is easy to see that if $T = T_\Lambda(X, Y)$ with $Y \neq X, \emptyset$, then μ_T has the congruence classes

(1) $[\xi] = \{p \in C_\Sigma(X) \mid \Lambda^\bullet(p) = \xi\}$,

(2) $[p_+] = \{p \in C_\Sigma(X) \mid \Lambda^\bullet(p) \in Y\}$, and

(3) $[p_-] = \{p \in C_\Sigma(X) \mid \Lambda^\bullet(p) \in X \setminus Y\}$.

Furthermore, $[\xi]$ is the identity element in $\mathrm{SM}(T)$ while $[p_+]$ and $[p_-]$ both are right zeros as $(p \cdot p_+, p_+)$, $(p \cdot p_-, p_-) \in \mu_T$ for any $p_+ \in [p_+]$, $p_- \in [p_-]$ and $p \in C_\Sigma(X)$. Hence the following corollary of Proposition 4.1.

Corollary 4.3. *For any $\Lambda \in pa(\Sigma)$, any leaf alphabet X, and any $T \in FProj_\Lambda(X)$, the syntactic monoid $\mathrm{SM}(T)$ is either trivial or isomorphic to the 3-element monoid $M = \{1, a, b\}$ in which 1 is the identity element and the elements a and b are right zeros.*

However, Corollary 4.3 does not mean that the Σ-VTL $FProj_\Lambda$ can be characterized by syntactic monoids. Indeed, the same syntactic monoids are obtained for every $\Lambda \in pa(\Sigma)$ and there are also completely different tree languages with syntactic monoids isomorphic to M.

For each X, $FProj_\Lambda(X)$ contains just one tree language for each $Y \subseteq X$, and hence $FProj_\Lambda(X)$ has $2^{|X|}$ elements. Let us consider the more general Σ-VTLs $FRA_\mathcal{L}$ ($\mathcal{L} \subseteq pa(\Sigma)$), i.e., the joins of the Σ-VTLs $FProj_\Lambda$. By the Variety Theorem, Propositions 3.3 and 3.1 translate into the following proposition about Σ-VTLs.

Proposition 4.2. *The Σ-VTLs $FRA_\mathcal{L}$ ($\mathcal{L} \subseteq pa(\Sigma)$) form a $2^{n(\Sigma)}$-element Boolean sublattice of the lattice of all Σ-VTLs. In this sublattice the least element is $Triv_\Sigma = \{\{\emptyset, T_\Sigma(X)\}\}_X$, the greatest element is FRA_Σ, and for all $\mathcal{L}, \mathcal{M} \subseteq pa(\Sigma)$, $FRA_\mathcal{L} \vee FRA_\mathcal{M} = FRA_{\mathcal{L}\cup\mathcal{M}}$, $FRA_\mathcal{L} \wedge FRA_\mathcal{M} = FRA_{\mathcal{L}\cap\mathcal{M}}$, and $FRA_\mathcal{L}^\complement = FRA_{pa(\Sigma)\setminus\mathcal{L}}$. The atoms of the sublattice, the Σ-VTLs $FProj_\Lambda$ ($\Lambda \in pa(\Sigma)$), are minimal Σ-VTLs.*

From Propositions 3.3 it also follows that for any $\mathcal{L} \subseteq pa(\Sigma)$ and any X, the members of $FRA_\mathcal{L}(X)$ are precisely the ΣX-tree languages saturated by $\rho_\mathcal{L}(X)$. Now, it is easy to see that, quite generally, if $\theta_1, \ldots, \theta_n (n \geq 1)$ are equivalences on a set U, then the subsets of U saturated by $\theta_1 \cap \ldots \cap \theta_n$ are precisely the sets that can be represented as finite unions of intersections $C_1 \cap \ldots \cap C_n$, where for each

$i \in [n]$, C_i is a subset of U saturated by θ_i. Hence the following description of the Σ-VTLs $FRA_{\mathcal{L}}$ is obtained by using Proposition 4.1. It could also be expressed by saying that $FRA_{\mathcal{L}}$ is the ring closure of $\bigcup_{\Lambda\in\mathcal{L}} FProj_\Lambda$.

Proposition 4.3. *For any set $\mathcal{L} = \{\Lambda_1, \ldots, \Lambda_n\} \subseteq pa(\Sigma)$ of projection alphabets and any X, the members of $FRA_{\mathcal{L}}(X)$ are precisely the unions of finitely many intersections*

$$T_{\Lambda_1}(X, Y_1) \cap \ldots \cap T_{\Lambda_n}(X, Y_n),$$

where $Y_1, \ldots, Y_n \subseteq X$.

The following decidability result is obvious by the finiteness of $\mathbf{FRA}_{\mathcal{L}}(X)$, but it also follows from the fact that a finite Σ-algebra belongs to $\mathbf{FRA}_{\mathcal{L}}$ iff it is isomorphic to a subdirect product of algebras $\mathcal{P}_\Lambda$ with $\Lambda \in pa(\Sigma)$.

Proposition 4.4. *For any $\mathcal{L} \subseteq pa(\Sigma)$, it can be decided whether a given regular ΣX-tree language belongs to $FRA_{\mathcal{L}}$.*

5 Comparisons with other varieties

It is to be expected that our varieties have little in common with most of the varieties of finite algebras or varieties of tree languages considered in the literature. Firstly, they are too small to contain other nontrivial varieties. In particular, the Σ-VTLs $FRA_{\mathcal{L}}$ contain no nonempty finite sets. On the other hand, the sets $T_\Lambda(X, Y)$ $(\emptyset \subset Y \subset X)$ are not defined by any local properties of their trees – as usually is the case. Let us make this incomparability explicit for a few Σ-VTLs. The precise definitions of these can be found in [12] or [13], for example, but the following informal descriptions should suffice here.

In the Σ-VTL $Nil_\Sigma = \{Nil_\Sigma(X)\}_X$, each set $Nil_\Sigma(X)$ consists of the finite and the co-finite ΣX-tree languages, and Nil^a is the Σ-VFA $\mathbf{Nil}_\Sigma$ of *nilpotent* (finite) Σ-algebras (defined in [4]).

A ΣX-tree language T is *definite* if there is a $k \geq 0$ such that the membership of a ΣX-tree in T depends only on the 'root segment' of t of height $k-1$. Similarly, T is *reverse definite* if there is a $k \geq 0$ such that whether or not $t \in T$ depends just on the subtrees of t of height $< k$. (In both cases, $k = 0$ means that no testing is needed and, accordingly, $T = \emptyset$ or $T = T_\Sigma(X)$.) By allowing combinations of these two types of tests, we get the *generalized definite* tree languages. The three Σ-VTLs obtained this way are denoted by Def_Σ, $RDef_\Sigma$ and $GDef_\Sigma$, and the corresponding Σ-VFAs by $\mathbf{Def}_\Sigma$, $\mathbf{RDef}_\Sigma$ and $\mathbf{GDef}_\Sigma$, respectively.

A *fork* of ΣX-tree is a configuration of the form $f(d_1, \ldots, d_m)$, where $f \in \Sigma_m$, $m > 0$ and $d_1, \ldots, d_m \in \Sigma \cup X$. A ΣX-tree language T is *local* if whether a ΣX-tree t belongs to T is determined by the set of forks appearing in t and its root label. The Σ-VTL Loc_Σ of *locally testable* Σ-tree languages is obtained as the Boolean closure of the Σ-family of local tree languages. Let $\mathbf{Loc}_\Sigma$ be the corresponding Σ-VFA.

Proposition 5.1. *For any set of path alphabets $\mathcal{L} \subseteq pa(\Sigma)$,*

(a) $FRA_{\mathcal{L}} \cap \mathcal{V} = Triv_\Sigma$ *if* $\mathcal{V}$ *is* $Nil_\Sigma, Def_\Sigma, RDef_\Sigma, GDef_\Sigma$ *or* Loc_Σ, *and*

(b) $\mathbf{FRA}_{\mathcal{L}} \cap \mathbf{K} = \mathbf{Triv}_\Sigma$ *if* $\mathbf{K}$ *is* $\mathbf{Nil}_\Sigma, \mathbf{Def}_\Sigma, \mathbf{RDef}_\Sigma, \mathbf{GDef}_\Sigma$ *or* $\mathbf{Loc}_\Sigma$.

Proof. By the Variety Theorem, assertions (a) and (b) are equivalent. The case $\mathcal{L} = \emptyset$ being trivial, we assume that $\mathcal{L} \neq \emptyset$.

Let us first assume that $\mathcal{L}$ consists of a single path alphabet Λ. Since $FProj_\Lambda$ is a minimal Σ-VTL and the intersection of any Σ-VTLs is a Σ-VTL, statement (a) holds for $\mathcal{L} = \{\Lambda\}$ if $FProj_\Lambda$ is not contained in any of the Σ-VTLs $\mathcal{V}$. To show this, we consider any $T_\Lambda(X,Y) \in FProj_\Lambda(X)$ such that $\emptyset \subset Y \subset X$. Since $T_\Lambda(X,Y)$ is neither finite nor co-finite, it is not in $Nil_\Sigma(X)$. To show that $T_\Lambda(X,Y)$ is not definite, we select any $f \in \Sigma$, $y \in Y$ and $x \in X \setminus Y$, and define two sequences of ΣX-trees by setting (1) $s_0 = y$, $t_0 = x$, and (2) $s_{n+1} = f(s_n, \ldots, s_n)$ and $t_{n+1} = f(t_n, \ldots, t_n)$ for all $n \geq 0$. Then, for every $k \geq 0$, the trees s_k and t_k have the same root segment of height $k-1$, but $s_k \in T_\Lambda(X,Y)$ while $t_k \notin T_\Lambda(X,Y)$, and therefore $T_\Lambda(X,Y)$ is not definite. Similar arguments can be used in the remaining cases. Since all the sets $T_\Lambda(X,Y)$ are recognized by $\mathcal{P}_\Lambda$, it follows that $\mathcal{P}_\Lambda$ cannot belong to any of the Σ-VFAs $\mathbf{Nil}_\Sigma, \mathbf{Def}_\Sigma, \mathbf{RDef}_\Sigma, \mathbf{GDef}_\Sigma$ or $\mathbf{Loc}_\Sigma$.

Consider now the general case $\emptyset \neq \mathcal{L} \subseteq pa(\Sigma)$. Let $\mathbf{K}$ be any one of the Σ-VFAs $\mathbf{Nil}_\Sigma$, $\mathbf{Def}_\Sigma$, $\mathbf{RDef}_\Sigma$, $\mathbf{GDef}_\Sigma$ or $\mathbf{Loc}_\Sigma$. Assume that $\mathbf{FRA}_{\mathcal{L}} \cap \mathbf{K}$ contains a nontrivial Σ-algebra $\mathcal{A}$. Then $\mathcal{A}$ would have a decomposition into a subdirect product of some subdirectly irreducible algebras $\mathcal{A}_1, \ldots, \mathcal{A}_n$ ($n \geq 1$) all of which belong to both $\mathbf{FRA}_{\mathcal{L}}$ and $\mathbf{K}$. However, the only nontrivial subdirectly irreducible algebras in $\mathbf{FRA}_{\mathcal{L}}$ are the algebras $\mathcal{P}_\Lambda$ with $\Lambda \in \mathcal{L}$, and by the first part of the proof, these do not belong to $\mathbf{K}$. Therefore we must have $\mathbf{FRA}_{\mathcal{L}} \cap \mathbf{K} = \mathbf{Triv}_\Sigma$. □

We conclude this section with two examples of Σ-VTLs that contain the Σ-VTLs $FRA_{\mathcal{L}}$. Thomas [15] calls a ΣX-tree language T *aperiodic* if there is an $n \geq 0$ such that

$$(\forall t \in T_\Sigma(X))(\forall p, q \in C_\Sigma(X))(t \cdot p^n \cdot q \in T \leftrightarrow t \cdot p^{n+1} \cdot q \in T).$$

The aperiodic Σ-tree languages form a Σ-VTL Ap_Σ that can be characterized by syntactic monoids [15].

Proposition 5.2. $FRA_\Sigma \subset Ap_\Sigma$.

Proof. We begin by showing that $FProj_\Lambda \subset Ap_\Sigma$ for every $\Lambda \in pa(\Sigma)$. Let $T = T_\Lambda(X,Y)$ be any set in $FProj_\Lambda(X)$. The remaining two cases being trivial, we may assume that $\emptyset \subset Y \subset X$. To show that for any $t \in T_\Sigma(X)$ and $p, q \in C_\Sigma(X)$, $t \cdot p \cdot q \in T$ iff $t \cdot p^2 \cdot q \in T$ (our "n" is 1), we distinguish two cases:

1. If $\Lambda^\bullet(q) \in X$, then $\Lambda^\bullet(t \cdot p \cdot q) = \Lambda^\bullet(q) = \Lambda^\bullet(t \cdot p^2 \cdot q)$, and hence $t \cdot p \cdot q \in T$ iff $t \cdot p^2 \cdot q \in T$.

2. If $\Lambda^\bullet(q) = \xi$, there are two subcases to consider. If $\Lambda^\bullet(p) \in X$, then $\Lambda^\bullet(t \cdot p \cdot q) = \Lambda^\bullet(p) = \Lambda^\bullet(t \cdot p^2 \cdot q)$, and hence $t \cdot p \cdot q \in T$ iff $t \cdot p^2 \cdot q \in T$. If $\Lambda^\bullet(p) = \xi$, then $\Lambda^\bullet(t \cdot p \cdot q) = \Lambda^\bullet(t) = \Lambda^\bullet(t \cdot p^2 \cdot q)$, and again $t \cdot p \cdot q \in T$ iff $t \cdot p^2 \cdot q \in T$.

The inclusion $FRA_\Sigma \subseteq Ap_\Sigma$ follows now because FRA_Σ is the join of Σ-VTLs contained in the Σ-VTL Ap_Σ. The inclusion is proper as all finite tree languages are aperiodic. □

Finally, let us consider the tree languages recognized by deterministic tree recognizers that read their input trees starting at the root and accepting at the leaves. General treatments of this topic and further references can be found in [5], [6], and [9].

A *deterministic top-down (DT)* Σ*-algebra* $\mathcal{B} = (B, \Sigma)$ consists of a non-empty set B and a Σ-indexed family of *top-down operations* $f^{\mathcal{B}}\colon B \longrightarrow B^m$ $(m \in r(\Sigma), f \in \Sigma_m)$. Such a DT Σ-algebra $\mathcal{B}$ is *finite* if B is a finite set. A *DT* ΣX*-recognizer* is a system $\mathbf{B} = (\mathcal{B}, b_0, \beta)$, where $\mathcal{B} = (B, \Sigma)$ is a finite DT Σ-algebra, the elements of which are called *states*, $b_0 \in B$ is the *initial state*, and $\beta : X \to \wp(B)$ is the *final state assignment*. If we extend β to a mapping $\beta_{\mathcal{B}} : T_\Sigma(X) \to \wp(B)$ by setting

(1) $\beta_{\mathcal{B}}(x) = \beta(x)$ for each $x \in X$, and

(2) $\beta_{\mathcal{B}}(t) = \{b \in B \mid f^{\mathcal{B}}(b) \in \beta_{\mathcal{B}}(t_1) \times \ldots \times \beta_{\mathcal{B}}(t_m)\}$ for $t = f(t_1, \ldots, t_m)$,

then for any $t \in T_\Sigma(X)$, $\beta_{\mathcal{B}}(t)$ is the set of the states $b \in B$ such that $\mathbf{B}$ reaches every leaf of t in an appropriate final state if started at the root of t in state b. Accordingly, the tree language *recognized* by $\mathbf{B}$ is defined as $T(\mathbf{B}) := \{t \in T_\Sigma(X) \mid b_0 \in \beta_{\mathcal{B}}(t)\}$. A ΣX-tree language T is said to be *DT-recognizable*, if $T = T(\mathbf{B})$ for a DT ΣX-recognizer $\mathbf{B}$. Let $DRec_\Sigma = \{DRec_\Sigma(X)\}_X$, where for each X, $DRec_\Sigma(X)$ is the set of all DT-recognizable ΣX-tree languages.[1] It is well known that $DRec_\Sigma$ is a proper subfamily of Rec_Σ.

Lemma 5.1. *$FProj_\Lambda \subset DRec_\Sigma$ for every $\Lambda \in pa(\Sigma)$.*

Proof. If $T \in FProj_\Lambda(X)$, then $T = T(\mathbf{A})$ for a ΣX-recognizer $\mathbf{A} = (\mathcal{P}_\Lambda, \alpha, F)$, where $F = \{1\}$. We define a DT ΣX-recognizer $\mathbf{B} = (\mathcal{B}, 1, \beta)$ as follows. Let $\mathcal{B} = (\{0, 1\}, \Sigma)$ be the DT Σ-algebra such that for any $m \in r(\Sigma)$, $f \in \Sigma_m$, $f^{\mathcal{B}}(0) = (0, \ldots, 0, \ldots, 0)$ and $f^{\mathcal{B}}(1) = (0, \ldots, 1, \ldots, 0)$, where the "1" is in position $i \in [m]$ if $f_i \in \Lambda$. For each $x \in X$, we set $\beta(x) = \{0, \alpha(x)\}$. By induction on $t \in T_\Sigma(X)$, it is easy to see that $\mathbf{B}$ reaches the leaf at the end of the path $\Lambda(t)$ in state 1 and all other leaves in state 0. Hence, $\mathbf{B}$ accepts t iff $\alpha(\Lambda^\bullet(t)) = 1$, i.e., iff $t \in T(\mathbf{A})$.

The inclusion is proper since every singleton tree language is DR-recognizable. □

Since $DRec_\Sigma$ is not a Σ-VTL, Lemma 5.1 does not imply that $FRA_\Sigma \subseteq DRec_\Sigma$. In fact, $FRA_\mathcal{L} \nsubseteq DRec_\Sigma$ if $\mathcal{L}$ contains two distinct projection alphabets Λ_1 and Λ_2. Indeed, if x and y are two different symbols in X, then

$$T := \{t \in T_\Sigma(X) \mid \Lambda_1^\bullet(t) = x, \Lambda_2^\bullet(t) = y \text{ or } \Lambda_1^\bullet(t) = y, \Lambda_2^\bullet(t) = x\}$$

[1] Note that in the literature DT-recognizable tree languages are often called DR-recognizable tree languages (derived from "root-to-frontier" instead of the currently dominating "top-down").

is recognized by $\mathcal{P}_{\Lambda_1} \times \mathcal{P}_{\Lambda_2}$ although it is not DT-recognizable. On the other hand, as shown by Jurvanen [8, 9], the Boolean closure $\mathfrak{B}DRec_\Sigma$ is the Σ-VTL generated by $DRec_\Sigma$, and hence Lemma 5.1 yields the following conclusion. That the inclusion is proper, is easily seen by considering, for example, any set of the form $\{f(x, \ldots, x)\}$.

Proposition 5.3. $FRA_\Sigma \subset \mathfrak{B}DRec_\Sigma$.

6 Solidity properties

Pöschel and Reichel [11] have shown that the rectangular Σ-algebras form the least nontrivial solid variety of Σ-algebras. For the general theory of solid varieties the reader may consult [10], for example. We shall consider the solidity properties of our Σ-VFAs and Σ-VTLs.

Let $\Xi := \{\xi_1, \xi_2, \xi_3, \ldots\}$ be a set of variables which do not appear in any of the other alphabets. For any $n \geq 1$, let $\Xi_n := \{\xi_1, \ldots, \xi_n\}$ and $T_\Sigma(\Xi_n)$ be the set of *n-ary Σ-terms*, and $T_\Sigma(\Xi) := \bigcup_{n \geq 1} T_\Sigma(\Xi_n)$ be the set of all *Σ-terms with variables*. If $t \in T_\Sigma(\Xi_n)$ and $t_1, \ldots, t_n$ are terms of any kind, then $t[t_1, \ldots, t_n]$ denotes the term obtained from t by substituting for every occurrence of a variable $\xi_1, \ldots, \xi_n$ the respective term $t_1, \ldots, t_n$. The term function $A^n \to A$ defined by an n-ary term $t \in T_\Sigma(\Xi_n)$ in a Σ-algebra $\mathcal{A} = (A, \Sigma)$ is denoted by $t^{\mathcal{A}}$.

A *hypersubstitution* of type Σ is a mapping $\varkappa : \Sigma \to T_\Sigma(\Xi)$ such that if $f \in \Sigma_m$, then $\varkappa(f) \in T_\Sigma(\Xi_m)$. Let $\mathcal{S}$ denote the set of all hypersubstitutions of type Σ. Any $\varkappa \in \mathcal{S}$ is extended to a mapping $\widehat{\varkappa} : T_\Sigma(\Xi) \to T_\Sigma(\Xi)$ by setting $\widehat{\varkappa}(\xi_i) = \xi_i$ for every $i \geq 1$, and $\widehat{\varkappa}(t) = \varkappa(f)[\widehat{\varkappa}(t_1), \ldots, \widehat{\varkappa}(t_m)]$ for $t = f(t_1, \ldots, t_m)$. We let $\varkappa$ denote $\widehat{\varkappa}$, too.

For any $\varkappa \in \mathcal{S}$ and any Σ-algebra $\mathcal{A} = (A, \Sigma)$, the Σ-algebra $\varkappa(\mathcal{A}) = (A, \Sigma)$ such that $f^{\varkappa(\mathcal{A})} = \varkappa(f)^{\mathcal{A}}$ for each $f \in \Sigma$, is a *derived algebra* of $\mathcal{A}$. In [7] it was noted that the solidity of varieties can be defined in terms of derived algebras, and the idea of solidity with respect to submonoids of the monoid of all hypersubstitutions of a given type was introduced in [2]. For a class $\mathcal{H} \subseteq \mathcal{S}$ of hypersubstitutions, a class $\mathbf{K}$ of Σ-algebras is said to be *$\mathcal{H}$-solid* if $\varkappa(\mathcal{A}) \in \mathbf{K}$ whenever $\mathcal{A} \in \mathbf{K}$ and $\varkappa \in \mathcal{H}$, and it is *solid* if it is $\mathcal{S}$-solid.

The first part of the following lemma can easily be verified by term induction. The second statement follows from the well-known fact that homomorphisms preserve also term functions.

Lemma 6.1. *Let $\varkappa$ be a hypersubstitution of type Σ, and let $\mathcal{A}$ and $\mathcal{B}$ be any Σ-algebras.*

(a) $t^{\varkappa(\mathcal{A})} = \varkappa(t)^{\mathcal{A}}$ *for any $n \geq 1$ and any n-ary term $t \in T_\Sigma(\Xi_n)$.*

(b) *Any homomorphism $\varphi : \mathcal{A} \to \mathcal{B}$ is also a homomorphism from $\varkappa(\mathcal{A})$ to $\varkappa(\mathcal{B})$.*

Let us call $\varkappa \in \mathcal{S}$ *permutative* if for all $m \in r(\Sigma)$ and $f \in \Sigma_m$, $\varkappa(f) = g(\xi_{i_1}, \ldots, \xi_{i_m})$ for some $g \in \Sigma_m$ and some permutation $(i_1, \ldots, i_m)$ of $(1, \ldots, m)$.

Let $p\mathcal{S}$ denote the set of all these hypersubstitutions. In the terminology of [14], they are precisely the linear, nondeleting, symbol-to-symbol $\Sigma\Sigma$-substitutions.

Proposition 6.1. *The Σ-VFAs* $\mathbf{Triv}_\Sigma$ *and* $\mathbf{FRA}_\Sigma$ *are solid, and* $\mathbf{FRA}_\Sigma$ *is the least nontrivial solid Σ-VFA. On the other hand, if $\emptyset \subset \mathcal{L} \subset pa(\Sigma)$, then* $\mathbf{FRA}_\mathcal{L}$ *is not even $p\mathcal{S}$-solid.*

Proof. Since the variety of rectangular Σ-algebras is solid by Theorem 5.1 of [11], also the Σ-VFA $\mathbf{FRA}_\Sigma$ is solid. Moreover, $\mathbf{FRA}_\Sigma \subseteq \mathbf{K}$ for every nontrivial solid Σ-VFA $\mathbf{K}$. Indeed, if $\mathcal{A} = (A, \Sigma)$ is any nontrivial member of $\mathbf{K}$, we obtain for any given $\Lambda \in pa(\Sigma)$ the Λ-projection algebra $\mathcal{B} = (A, \Sigma)$ as the derived algebra $\varkappa(\mathcal{A})$, if for any $m \in r(\Sigma)$ and $f \in \Sigma_m$, we set $\varkappa(f) = \xi_i$ for the $i \in [m]$ such that $f_i \in \Lambda$. From this it follows that $\mathbf{K}$ contains all the algebras $\mathcal{P}_\Lambda$ with $\Lambda \in pa(\Sigma)$, and hence all of $\mathbf{FRA}_\Sigma$.

Assume now that $\emptyset \subset \mathcal{L} \subset pa(\Sigma)$. To prove that $\mathbf{FRA}_\mathcal{L}$ is not $p\mathcal{S}$-solid, it obviously suffices to show that for any two projection alphabets $\Lambda, \Lambda' \in pa(\Sigma)$, there exists a permutative hypersubstitution $\varkappa$ for which $\mathcal{P}_{\Lambda'} = \varkappa(\mathcal{P}_\Lambda)$. To define such a $\varkappa$, consider any $m \in r(\Sigma)$ and $f \in \Sigma_m$. If $f_i \in \Lambda$ and $f_j \in \Lambda'$ $(i, j \in [m])$, then we set $\varkappa(f) = f(\xi_{i_1}, \ldots, \xi_{i_m})$, where $(i_1, \ldots, i_m)$ is the permutation of $(1, \ldots, m)$ that just transposes i and j. It is then clear that $f^{\varkappa(\mathcal{P}_\Lambda)} = f^{\mathcal{P}_{\Lambda'}}$. □

To define the solidity of Σ-VTLs, we adapt some notions from [14] to the present setting of a fixed ranked alphabet. Firstly, if $\mathcal{H} \subseteq \mathcal{S}$ is a class of hypersubstitutions of type Σ, an *$\mathcal{H}$-morphism* $\varphi : T_\Sigma(X) \to T_\Sigma(Y)$ is defined by its *underlying hypersubstitution* $\varphi_* \in \mathcal{H}$ and a mapping $\varphi_X : X \to T_\Sigma(Y)$ as follows:

(1) $x\varphi = \varphi_X(x)$ for $x \in X$;

(2) $t\varphi = \varphi_*(f)[t_1\varphi, \ldots, t_m\varphi]$ for $t = f(t_1, \ldots, t_m)$.

The following fact is easily verified.

Lemma 6.2. *If $\varphi : T_\Sigma(X) \to T_\Sigma(Y)$ is an $\mathcal{S}$-morphism, then $\varphi : \mathcal{T}_\Sigma(X) \to \varphi_*(\mathcal{T}_\Sigma(Y))$ is a homomorphism of Σ-algebras.*

For any $\mathcal{H} \subseteq \mathcal{S}$, a Σ-VTL $\mathcal{V} = \{\mathcal{V}(X)\}_X$ is *$\mathcal{H}$-solid* if $T \in \mathcal{V}(Y)$ implies that $T\varphi^{-1} \in \mathcal{V}(X)$ for every $\mathcal{H}$-morphism $\varphi : T_\Sigma(X) \to T_\Sigma(Y)$. In particular, $\mathcal{V}$ is *solid* if it is $\mathcal{S}$-solid.

In [14] it was shown that any general variety of finite algebras and the corresponding general variety of tree languages (the "general" signifies that the ranked alphabet is not fixed) have matching solidity properties. Although restricting such general varieties to one given ranked alphabet does not yield exactly our Σ-VFAs and Σ-VTLs, this holds also here.

Lemma 6.3. *Let $\mathcal{H}$ be any class of hypersubstitutions of type Σ. If a Σ-VFA $\mathbf{K}$ is $\mathcal{H}$-solid, then so is the Σ-VTL $\mathbf{K}^t$.*

Proof. Consider any $\mathcal{H}$-morphism $\varphi : T_\Sigma(X) \to T_\Sigma(Y)$. If $T \in \mathbf{K}^t(Y)$, then there exist an algebra $\mathcal{A} = (A, \Sigma)$ in $\mathbf{K}$ and a homomorphism $\psi : \mathcal{T}_\Sigma(Y) \to \mathcal{A}$ such that

$T = F\psi^{-1}$ for some $F \subseteq A$. By Lemmas 6.2 and 6.1, $\varphi : \mathcal{T}_\Sigma(X) \to \varphi_*(\mathcal{T}_\Sigma(Y))$ and $\psi : \varphi_*(\mathcal{T}_\Sigma(Y)) \to \varphi_*(\mathcal{A})$ are homomorphisms. Hence $\varphi\psi : \mathcal{T}_\Sigma(X) \to \varphi_*(\mathcal{A})$ is a homomorphism and $T\varphi^{-1} = F(\varphi\psi)^{-1}$. Since $\varphi_*(\mathcal{A}) \in \mathbf{K}$, this means that $T\varphi^{-1} \in \mathbf{K}^t(X)$. □

Proving the converse of Lemma 6.3 would require some further preparations, so we avoid its use and just refer the reader to [14] for the corresponding fact about general varieties.

Proposition 6.2. *The Σ-VTLs $Triv_\Sigma$ and FRA_Σ are solid, and FRA_Σ is the least nontrivial Σ-VTL. However, if $\emptyset \subset \mathcal{L} \subset pa(\Sigma)$, then $FRA_\mathcal{L}$ is not even $p\mathcal{S}$-solid.*

Proof. That $Triv_\Sigma$ and FRA_Σ are solid follows from Proposition 6.1 by Lemma 6.3.

To show that $FRA_\mathcal{L}$ is not $p\mathcal{S}$-solid for $\emptyset \subset \mathcal{L} \subset pa(\Sigma)$, consider any $\Lambda, \Lambda' \in pa(\Sigma)$ such that $\Lambda \in \mathcal{L}$, but $\Lambda' \notin \mathcal{L}$. Let $X = \{x, y\}$ and $\varphi : T_\Sigma(X) \to T_\Sigma(X)$ be the $p\mathcal{S}$-morphism such that φ_* is the hypersubstitution $\nwarrow$ defined in the second part of the proof of Proposition 6.1, and $\varphi_X(x) = x$ and $\varphi_X(y) = y$. Then $T_\Lambda(X, \{y\}) \in FRA_\mathcal{L}(X)$, but $T_\Lambda(X, \{y\})\varphi^{-1} = T_{\Lambda'}(X, \{y\}) \notin FRA_\mathcal{L}(X)$. □

References

[1] S. Burris and H.P. Sankappanavar, *A Course in Universal Algebra*. Springer-Verlag, New York 1981.

[2] K. Denecke and M. Reichel, Monoids of hypersubstitutions and M-solid varieties. In: *Contributions to General Algebra 9* (ed. G. Pilz), Hölder-Pichler-Tempsky, Wien-Stuttgart 1995, 117–126.

[3] Z. Ésik, A variety theorem for trees and theories, *Publicationes Mathematicae Debrecen* **54** Supplementum (1999), 711–762.

[4] F. Gécseg and B. Imreh, On a special class of automata. In: *Automata, Languages and Programming Systems*, Proc. Conf. Salgótárjan May 23–26, 1988 (eds. F. Gécseg and I. Peák), Department of Mathematics, Karl Marx University of Economics, Budapest 1988, 141–152.

[5] F. Gécseg and M. Steinby, *Tree Automata*, Akadémiai Kiadó, Budapest 1984.

[6] F. Gécseg and M. Steinby, Tree languages. In: *Handbook of Formal Languages*, Vol. 3 (eds. G. Rozenberg and A. Salomaa), Springer-Verlag, Berlin 1997, 1–69.

[7] E. Graczyńska and D. Schweigert, Hypervarieties of a given type, *Algebra Universalis* **27** (1990), 303–318.

[8] E. Jurvanen, The Boolean closure of DR-recognizable tree languages, *Acta Cybernetica* **10** (1992), 255–272.

[9] E. Jurvanen, *On tree languages defined by deterministic root-to-frontier recognizers*, PhD thesis, Department of Mathematics, University of Turku, Turku 1995.

[10] J. Koppitz and K. Denecke, *M-Solid Varieties of Algebras*, Springer Science+Business Media, New York 2006.

[11] R. Pöschel and M. Reichel, Projection algebras and rectangular algebras. In: *General Algebra and Applications* (eds. K. Denecke and H. -J. Vogel), Heldermann Verlag, Berlin 1993, 180–194.

[12] M. Steinby: A theory of tree language varieties. In: *Tree Automata and Languages* (eds. M. Nivat and A. Podelski), North-Holland, Amsterdam 1992, 57–81.

[13] M. Steinby, Algebraic classifications of regular tree languages. In: *Structural Theory of Automata, Semigroups, and Universal Algebra* (eds. V.B. Kudryavtsev and I.G. Rosenberg), Springer, Dordrecht 2005, 381–432.

[14] M. Steinby, On the solidity of general varieties of tree languages, *Discussiones Mathematicae. General Algebra and Applications* **32** (2012), 23–53.

[15] W. Thomas, Logical aspects in the study of tree languages. In: *9th Colloquium on Trees in Algebra and Programming* (Proc. 9th CAAP, Bordeaux 1984, ed. B. Courcelle), Cambridge University Press, London 1984, 31–49.

Received 14th June 2015

Acta Cybernetica **22** (2015) 517–551.

On Ground Word Problem of Term Equation Systems

Sándor Vágvölgyi*

To the memory of my teacher and colleague Ferenc Gécseg

Abstract

We give semi-decision procedures for the ground word problem of variable preserving term equation systems and term equation systems. They are natural improvements of two well known trivial semi-decision procedures. We show the correctness of our procedures.

Keywords: term equation systems; ground word problem; Knuth-Bendix completion procedure; ground term rewriting systems

1 Introduction

A term equation $l \approx r$ is called variable preserving if the same variables occur in the left-hand side l as in the right-hand side r. A term equation system (TES) E is called variable preserving if all of its equations are variable preserving. The ground word problem is undecidable even for variable-preserving TESs, see Example 4.1.4 on page 60 in [1]. We recall the well known trivial semi-decision procedure *PRO1* for the ground word problem of variable preserving TESs and its straightforward generalization, the trivial semi-decision procedure *PRO2* for the ground word problem of TESs.

On the basis of *PRO1*, we give a semi-decision procedure *PRO3* for the ground word problem of variable preserving TESs. Given a TES E and ground terms p, q over the ranked alphabet Σ, procedure *PRO3* constructs the ground TESs (GTESs) P_i and Q_i, $i \geq 1$ such that

(*a*) $P_i \cup Q_i \subseteq \leftrightarrow_E^*$ for $i \geq 1$.

Condition (a) ensures that the congruence closure of $P_i \cup Q_i$ is a subset of $\leftrightarrow_E^*$. Procedure *PRO3* outputs an answer and halts if and only if

(*b*) there is a $j \geq 1$ such that

$p \leftrightarrow_{P_j \cup Q_j}^* q$ or

$\leftrightarrow_{P_j}^* \cap (\{ p \} \times T_\Sigma) = \leftrightarrow_E^* \cap (\{ p \} \times T_\Sigma)$ or

*Department of Foundations of Computer Science, University of Szeged, Árpád tér 2, H-6720 Szeged, Hungary. E-mail: vagvolgy@inf.u-szeged.hu

$\leftrightarrow^*_{Q_j} \cap(\{\, q \,\} \times T_\Sigma) = \leftrightarrow^*_E \cap(\{\, q \,\} \times T_\Sigma)$.
Condition (b) says that we have a proof of $p \leftrightarrow^*_E q$, or the intersection of $\leftrightarrow^*_{P_j}$ with $(\{\, p \,\} \times T_\Sigma)$ is equal to that of $\leftrightarrow^*_E$, or the intersection of $\leftrightarrow^*_{Q_j}$ with $(\{\, q \,\} \times T_\Sigma)$ is equal to that of $\leftrightarrow^*_E$. Assume that (b) holds. If $p \leftrightarrow^*_{P_j \cup Q_j} q$ holds, *PRO3* outputs 'yes', and halts. Otherwise, if

- the intersection of $\leftrightarrow^*_{P_j}$ with $(\{\, p \,\} \times T_\Sigma)$ is equal to that of $\leftrightarrow^*_E$, or
- the intersection of $\leftrightarrow^*_{Q_j}$ with $(\{\, q \,\} \times T_\Sigma)$ is equal to that of $\leftrightarrow^*_E$,

then $p \leftrightarrow^*_E q$ does not hold either. Hence semi-decision procedure *PRO3* outputs 'no' and halts.

Procedure *PRO3* constructs the ground TESs (GTESs) P_i and Q_i, $i \geq 1$ in the following way. We put a ground instance $l' \approx r'$ of an equation $l \approx r$ of $E \cup E^{-1}$ in P_1 if l' is a subterm of p. Then we iterate the following computation items.

- We convert the GTES P_i into an equivalent reduced ground term rewrite system R_i applying Snyder's fast ground completion algorithm [19].
- We define the GTES P_{i+1} from the reduced ground term rewrite system R_i by adding all ground instances $l \approx r$ of equations in $E \cup E^{-1}$ such that

- $l \approx r$ is not in $\leftrightarrow^*_{P_i}$ and that

- there exists a term s such that the conversion $p \leftrightarrow^*_{P_i} s$ can be continued applying $l \approx r$ to s. If $P_{i+1} = R_i$, then we let $R_{i+1} = R_i$, and hence $R_i = P_j = R_j$ holds for $j \geq i + 1$.

Here we consider both the reduced ground term rewrite system R_i and the GTES P_{i+1} as subsets of $T_\Sigma \times T_\Sigma$. Furthermore, we consider a ground instance of an equation in $E \cup E^{-1}$ as an element of $T_\Sigma \times T_\Sigma$.

We define the GTES Q_i symmetrically to P_i for $i \geq 1$.

Procedure *PRO3* computes in the following way. For each $i = 1, 2, \ldots$,

- if $p \leftrightarrow^*_{P_i \cup Q_i} q$, then we output the answer 'yes' and halt;
- otherwise, if $i \geq 2$ and we did not add ground instances of equations in $E \cup E^{-1}$ to the reduced ground term rewrite system $R_{P_{i-1}}$, equivalent to P_{i-1}, or to the reduced ground term rewrite system R_{Q_i}, equivalent to Q_{i-1}, in the previous iteration step, then we output the answer 'no' and halt.

Assume that $p \leftrightarrow^*_E q$. Then, at some step during the run of procedure *PRO3*, $p \leftrightarrow^*_{P \cup Q} q$ becomes true, and procedure *PRO3* outputs 'yes' and halt. If $p \leftrightarrow^*_E q$ does not hold, then procedure *PRO3* either outputs 'no' and halts or runs forever.

We give a semi-decision procedure *PRO4* for the ground word problem of TESs. We obtain it generalizing *PRO3* taking into account *PRO2*. The main difference is the following. We define P_{i+1} from R_i by adding all ground instances $l' \approx r'$ of the equations $l \approx r$ in $E \cup E^{-1}$ such that

- $l' \approx r'$ is not in $\leftrightarrow^*_{P_i}$, that
- there exists a term s such that a conversion $p \leftrightarrow^*_{P_i} s$ can be continued applying $l \approx r$ to s, and that
- we substitute some finitely many ground terms depending on i, R_i, and p, for those variables in r that do not appear in l.

We modify the halting condition of the proceedure so that it stops if we did not add ground instances of equations in $E \cup E^{-1}$ to P_i or Q_i in two successive iteration

steps. We need two successive steps rather than one. Because, in general, the heights of the substituted terms becomes larger in each step. If we do not add ground equations to P_i in a step, then in the next step we still may add ground equations to P_i.

Procedures *PRO3* and *PRO4* compute in a different way than all versions of the Knuth-Bendix completion procedure. To some instances of the ground word problem of a TES E, procedures *PRO3* and *PRO4* give an answer sooner than all versions of the Knuth-Bendix completion procedure or it is open whether some version of the Knuth-Bendix completion procedure gives an answer at all. Consequently, they may compute efficiently for some instances of the ground word problem of a TES E, when the various versions of the Knuth-Bendix completion procedure does not give an answer to the ground word problem of a TES E at all or at least not in a reasonable time. However, it is still open in which cases are *PRO3* and *PRO4* really efficient.

In Section 2, we present a brief review of the notions, notations, and preliminary results used in the paper. In Section 3 we introduce and study the concept of reading-up reachability for reduced ground term rewriting systems. In Section 4 we present the procedures *PRO1* and *PRO2*. In Section 5, we present the procedure *PRO3*, and show its correctness. We give examples when procedure *PRO3* is more efficient than procedure *PRO1*. In Section 6, we present the procedure *PRO4*, and show its correctness. In Section 7, we compare procedures *PRO3* and *PRO4* with the basic Knuth-Bendix completion procedure (see Section 7.1 in [1]), an improved version of the Knuth-Bendix completion procedure described by a set of inference rules (see Section 7.2 in [1]), the goal-directed completion procedure based on SOUR graphs [13, 14], and the unfailing Knuth-Bendix completion procedure [2]. In Section 8, we sum up our results, and explain the applicability of procedures *PRO3* and *PRO4*.

2 Preliminaries

In this section we present a brief review of the notions, notations and preliminary results used in the paper. For all unexplained notions and notation see [1].

Relations. Let ρ be an equivalence relation on A. Then for every $a \in A$, we denote by a/ρ the ρ-class containing a, i.e. $a/\rho = \{\, b \mid a\rho b \,\}$. For each $B \subseteq A$, let $B/\rho = \{\, b/\rho \mid b \in B \,\}$.

2.1 Abstract Reduction Systems

An abstract reduction system is a pair $(A, \to)$, where the reduction $\to$ is a binary relation on the set A. $\to^{-1}$, $\leftrightarrow$, $\to^*$, and $\leftrightarrow^*$ denote the inverse, the symmetric closure, the reflexive transitive closure, and the reflexive transitive symmetric closure of the binary relation $\to$, respectively.

- $x \in A$ is reducible if there is y such that $x \to y$.
- $x \in A$ is irreducible if it is not reducible.

• $y \in A$ is a normal form of $x \in A$ if $x \to^* y$ and y is irreducible. If $x \in A$ has a unique normal form, the latter is denoted by $x{\downarrow}$.

• $y \in A$ is a descendant of $x \in A$ if $x \to^* y$.

• $x \in A$ and $y \in A$ are joinable if there is a z such that $x \to^* z \leftarrow^* y$, in which case we write $x{\downarrow}y$.

The reduction $\to$ is called

• confluent if for all $x, y_1, y_2 \in A$, if $y_1 \leftarrow^* x \to^* y_2$, then $y_1{\downarrow}y_2$;

• locally confluent if for all $x, y_1, y_2 \in A$, if $y_1 \leftarrow x \to y_2$, then $y_1{\downarrow}y_2$;

• terminating if there is no infinite chain $x_0 \to x_1 \to x_2 \to \cdots$;

• convergent if it is both confluent and terminating.

If $\to$ is convergent, then each $x \in A$ has a unique normal form [1].

Terms. A ranked alphabet Σ is a finite set of symbols in which every element has a unique rank in the set of nonnegative integers. For each integer $m \geq 0$, Σ_m denotes the elements of Σ which have rank m.

Let Y be a set of variables. The set of terms over Σ with variables in Y is denoted by $T_\Sigma(Y)$. The set $T_\Sigma(\emptyset)$ is written simply as T_Σ and called the set of ground terms over Σ. We specify a countably infinite set $X = \{ x_1, x_2, \dots \}$ of variables which will be kept fixed in this paper. Moreover, we put $X_n = \{ x_1, x_2, \dots, x_n \}$, for $n \geq 0$. Hence $X_0 = \emptyset$. For any $i \geq 1$ and $j \geq 0$, let $X_{[i,j]} = \emptyset$ if $i > j$, and let $X_{[i,j]} = \{ x_i, x_{i+1}, \dots, x_j \}$ otherwise.

For a term $t \in T_\Sigma(X)$, the height $height(t) \in N$ is defined by recursion:

(a) if $t \in \Sigma_0 \cup X$, then $height(t) = 0$,

(b) if $t = \sigma(t_1, \dots, t_m)$ with $m \geq 1$ and $\sigma \in \Sigma_m$, then
$height(t) = 1 + max(height(t_i) \mid 1 \leq i \leq m)$.

For each $k \geq 0$, $HE_{\Sigma,\leq k}(X) = \{ t \in T_\Sigma(X) \mid height(t) \leq k \}$.

Let N be the set of all positive integers. N^* stands for the free monoid generated by N with empty word λ as identity element. For each word $\alpha \in N^*$, $length(\alpha)$ stands for the length of α. Consider the words $\alpha, \beta, \gamma \in N^*$ such that $\alpha = \beta\gamma$. Then we say that β is a prefix of α. Furthermore, if $\alpha \neq \beta$, then β is a proper prefix of α. For a term $t \in T_\Sigma(X)$, the set $Pos(t) \subseteq N^*$ of positions is defined by recursion:

(i) if $t \in \Sigma_0 \cup X$, then $Pos(t) = \{ \lambda \}$, and

(ii) if $t = \sigma(t_1, \dots, t_m)$ with $m \geq 1$ and $\sigma \in \Sigma_m$, then $Pos(t) = \{ \lambda \} \cup \{ i\alpha \mid 1 \leq i \leq m \text{ and } \alpha \in Pos(t_i) \}$.

For each term $t \in T_\Sigma(X)$, $size(t)$ is the cardinality of $Pos(t)$.

For each $t \in T_\Sigma(X)$ and $\alpha \in Pos(t)$, we introduce the subterm $t/\alpha \in T_\Sigma(X)$ of t at α as follows:

(a) for $t \in \Sigma_0 \cup X$, $t/\lambda = t$;

(b) for $t = \sigma(t_1, \dots, t_m)$ with $m \geq 1$ and $f \in \Sigma_m$, if $\alpha = \lambda$ then $t/\alpha = t$, otherwise, if $\alpha = i\beta$ with $1 \leq i \leq m$, then $t/\alpha = t_i/\beta$

For any $t \in T_\Sigma(X)$, $\alpha \in Pos(t)$, and $r \in T_\Sigma(X)$, we define $t[\alpha \leftarrow r] \in T_\Sigma(X)$.

(i) If $\alpha = \lambda$, then $t[\alpha \leftarrow r] = r$.

(ii) If $\alpha = i\beta$, for some integer i, then $t = \sigma(t_1, \ldots, t_m)$ with $f \in \Sigma_m$ and $1 \leq i \leq m$. Then $t[\alpha \leftarrow r] = \sigma(t_1, \ldots, t_{i-1}, t_i[\beta \leftarrow r], t_{i+1}, \ldots, t_m)$.

For a term $t \in T_\Sigma(X)$, the set $sub(t)$ of subterms of t is defined as $sub(t) = \{\, t/\alpha \mid \alpha \in Pos(t) \,\}$.

Given a term $t \in T_\Sigma(X_n)$, $n \geq 0$, and terms $t_1, \ldots, t_n$, we denote by $t[t_1, \ldots, t_n]$ the term which can be obtained from t by replacing each occurrence of x_i in t by t_i for $1 \leq i \leq n$. A context is a term $u \in T_{\Sigma \cup \{\diamond\}}$, where the nullary symbol $\diamond$ appears exactly once in u. We denote the set of all contexts over Σ by C_Σ. For a context u and a term t, $u[t]$ is defined from u by replacing the occurrence of $\diamond$ with t.

For the sake of simplicity, we may write unary terms as strings. For example, we write $fgh\#$ for the term $f(g(h(\#)))$ and f^3x_1 for $f(f(f(x_1)))$, where f, g, h are unary symbols and $\#$ is a nullary symbol.

Algebras. Let Σ be a ranked alphabet. A Σ algebra is a system $\mathbf{B} = (B, \Sigma^{\mathbf{B}})$, where B is a nonempty set, called the carrier set of $\mathbf{B}$, and $\Sigma^{\mathbf{B}} = \{\, f^{\mathbf{B}} \mid f \in \Sigma \,\}$ is a Σ-indexed family of operations over B such that for every $f \in \Sigma_m$ with $m \geq 0$, $f^{\mathbf{B}}$ is a mapping from B^m to B. An equivalence relation $\rho \subseteq B \times B$ is a congruence on $\mathbf{B}$ if

$$f^{\mathbf{B}}(t_1, \ldots, t_m) \rho f^{\mathbf{B}}(p_1, \ldots, p_m)$$

whenever $f \in \Sigma_m$, $m \geq 0$, and $t_i \rho p_i$, for $1 \leq i \leq m$. For each $B' \subseteq B$, let $[B']_\rho = \{\, [b]_\rho \mid b \in B' \,\}$. In this paper we shall mainly deal with the algebra $\mathbf{TA} = (T_\Sigma, \Sigma)$ of ground terms over Σ, where for any $f \in \Sigma_m$ with $m \geq 0$ and $t_1, \ldots, t_m \in T_\Sigma$, we have

$$f^{\mathbf{TA}}(t_1, \ldots, t_m) = f(t_1, \ldots, t_m)\,.$$

We now recall the concept of a set of representatives for a congruence ρ and a set of ρ-classes.

Definition 1. [6] Let ρ be a congruence on $\mathbf{TA}$ and let A be a set of ρ-classes. A set REP of ground terms is called a set of *representatives* for A if

- $REP \subseteq \bigcup A$,
- $\bigcup(\, sub(t) \mid t \in REP \,) \subseteq REP$, and
- each class $Z \in A$ contains exactly one term $t \in REP$.

Term equation systems. Let Σ be a ranked alphabet. A term equation system (TES for short) E over Σ is a finite subset of $T_\Sigma(X) \times T_\Sigma(X)$. Elements (l, r) of E are called equations and are denoted by $l \approx r$. The reduction relation $\rightarrow_E \subseteq T_\Sigma(X) \times T_\Sigma(X)$ is defined as follows. For any terms $s, t \in T_\Sigma(X)$, $s \rightarrow_E t$ if there is a pair $l \approx r$ in E and a context $u \in C_\Sigma(X_1)$ and a substition δ such that $s = u[\delta(l)]$ and $t = u[\delta(r)]$. When we apply an arbitrary equation $l \approx r \in E \cup E^{-1}$, we rename the variables of l and r such that $l \in T_\Sigma(X_{k+m})$ and $r \in T_\Sigma(X_k \cup X_{[k+m+1, k+m+\ell]})$ for some $k, m, \ell \geq 0$.

The word problem for a TES E is the problem of deciding for arbitrary $p, q \in T_\Sigma(X)$ whether $p \leftrightarrow_E^* q$. The ground word problem for E is the word problem restricted to ground terms p and q.

For the notion of a term rewriting system (TRS), see Section 4.2 in [1]

Knuth-Bendix completion procedure. We now briefly recall the basic Knuth-Bendix completion procedure, see Section 7.1 in [1]. The basic Knuth-Bendix completion procedure starts with a TES E and tries to find a convergent TRS R that is equivalent to E. A reduction order $>$ is provided as an input for the procedure. Since the word problem is not decidable in general, a finite convergent TRS cannot always be obtained. In the basic Knuth-Bendix completion procedure this could be due to failure or to non-termination of completion. In the initialization phase, the basic completion procedure removes trivial identities of the form $s = s$ and tries to orient the remaining nontrivial identities. If this succeeds, then it computes all critical pairs of the TRS obtained. The terms in each critical pair $\langle s, t\rangle$ are reduced to their normal forms $\hat{s}$ and $\hat{t}$. If the normal forms are identical, then this critical pair is joinable, and nothing needs to be done for it. Otherwise, the procedure tries to orient the terms $\hat{s}$ and $\hat{t}$ into the rewrite rule $\hat{s} \to \hat{t}$ with $\hat{s} > \hat{t}$ or $\hat{t} \to \hat{s}$ with $\hat{t} > \hat{s}$. In this way the procedure orients all instances of the terms $\hat{s}$ and $\hat{t}$ as well. If this succeeds, then the new rule is added to the current rewrite system. This process is iterated until failure occurs or the rewrite system is not changed during a step of the iteration, that is, the system does not have non-joinable critical pairs.

If the basic completion procedure applied to $(E, >)$ terminates succesfully with output R, then R is a finite convergent TRS that is equivalent to E. In this case, R yields a decision procedure for the word problem for E. If the basic completion procedure applied to $(E, >)$ does not terminate, then it outputs an infinite convergent TRS that is equivalent to E. In this case, the completion procedure can be used as a semidecision procedure for the word problem for E.

Assume that we want to decide for given terms $p, q \in T_\Sigma(X)$, whether $p \leftrightarrow_E^* q$ holds. We call the pair (p, q) the *goal*. The basic Knuth-Bendix completion procedure is independent of the goal. Hence, if $p \leftrightarrow_E^* q$ does not hold, and the set E of equations has no finite convergent system, then the basic Knuth-Bendix completion will run forever. In the light of this observation, Lynch and Strogova [13, 14] presented a goal-directed completion procedure based on SOUR graphs. Similarly to the basic Knuth-Bendix completion procedure, the goal-directed completion procedure uses a reduction order $>$. Unlike the basic Knuth-Bendix completion procedure, it uses some inference rules. The main difference, described in an intuitive simplified way, is the following. Along the completion procedure, we try to construct a rewrite system R and a conversion

$$p = r_1 \underset{R}{\leftrightarrow} r_2 \underset{R}{\leftrightarrow} \cdots \underset{R}{\leftrightarrow} r_n = q,\ n \geq 1 \tag{1}$$

in a nondeterministic way. We compute and orient critical pairs and control the completion process keeping in our mind that the rules of R should be applicable along a conversion (1). When orienting the equations into rules along the comple-

tion process, we do not put a rule in R if it is not applicable along a conversion (1). If we do not find a conversion (1), the goal-directed completion procedure detects that $(p, q) \notin \leftrightarrow_E^*$, outputs 'no' and halts. Consider the following example. Let ranked alphabet Σ consist of the unary symbols f, g and the nullary symbols \$, #. Consider the variable preserving TES $E = \{\, fffx \approx gfx \,\}$. We raise the problem whether $\$ \leftrightarrow_E^* \#$. The basic Knuth-Bendix completion procedure runs forever on this example [13]. Along the goal oriented completion procedure, we find no rewrite rule such that it is applicable along a conversion $\$ = r_1 \leftrightarrow_R r_2 \leftrightarrow_R \cdots \leftrightarrow_R r_n = \#$, $n \geq 1$. Therefore, the goal-directed completion procedure detects that $(\$, \#) \notin \leftrightarrow_E^*$, outputs 'no', and halts [13].

We now adopt a more detailed description of the goal-directed completion procedure. [14] The goal-directed completion procedure uses a reduction order $>$ and computes critical pairs equipped with equational and ordering constraints, and constructs a graph. "The goal-directed completion procedure has two phases. The first phase is the compilation phase. In this phase, all the edges and the recursive constraints labelling each edge are created. This phase also takes into account the goal to be solved. Importantly, this phase takes only polynomial time, because there are only polynomially many edges in the graph. The result of this phase is a constrained tree automaton representing a schematized version of the completed system, and a set of constraints representing potential solutions to the goal. The constraints that are generated are the equational constraints representing the unification problems, and ordering constraints arising from the critical pair inferences.

The second phase is the goal solving (or constraint solving) phase. In this phase, the potential solutions to the goal are solved in order to determine whether they are actual solutions of the goal. This phase can take infinitely long, since the constraints are recursive. Step by step a constraint is rolled back, based on which edges it is created from, and the equational and ordering constraints are solved along the way. In some cases, the ordering constraints cause the recursion to halt, and therefore the constraints are completely solved. The procedure is truly goal oriented, because only a polynomial amount of time is spent compiling the set of equations. The rest of the time is spent working backwards from the goal to solve the constraints. If the procedure is examined more closely, we see that the second phase of the procedure is exactly a backwards process of completion. A schematization of an equation in the completed system is applied to the goal, step by step until it rewrites to an identity. At the same time, the schematized equation that is selected is worked backwards until we reach the original equations from which it is formed." [14]

See Section 7.2 in [1] for an improved version of the Knuth-Bendix completion procedure described by a set of inference rules. A detailed description of the unfailing Knuth-Bendix completion procedure can be found in [2].

Ground term equation systems and rewriting systems. A ground term equation system (GTES) E over a ranked alphabet Σ is a finite binary relation on T_Σ. Elements (l, r) of E are called equations and are denoted by $l \approx r$. The reduction relation $\rightarrow_E \subseteq T_\Sigma(X) \times T_\Sigma(X)$ is defined as follows. For any ground terms $s, t \in T_\Sigma$, $s \rightarrow_E t$ if there is a pair $l \approx r$ in E and a context $u \in C_\Sigma(X_1)$

such that $s = u[l]$ and $t = u[r]$. It is well known that the relation $\leftrightarrow_E^*$ is a congruence on the term algebra **TA** [18]. We call $\leftrightarrow_E^*$ the congruence induced by E. The size of E is defined as the number of occurrences of symbols in the set. $sub(E) = \{\, sub(l) \mid l \approx r \in E \cup E^{-1} \,\}$. Clearly, $\leftrightarrow_E^* \cap (sub(E) \times sub(E))$ is an equivalence relation on $sub(E)$. The word problem for a GTES E is the problem of deciding for arbitrary $p, q \in T_\Sigma$ whether $p \leftrightarrow_E^* q$.

A ground term rewrite system (GTRS) over a ranked alphabet Σ is a finite subset R of $T_\Sigma \times T_\Sigma$. The elements of R are called rules and a rule $(l, r) \in R$ is written in the form $l \to r$ as well. Moreover, we say that l is the left-hand side and r is the right-hand side of the rule $l \to r$. $lhs(R) = \{l \mid l \to r \in R\}$, $rhs(R) = \{r \mid l \to r \in R\}$. $sub(R) = \{\, sub(l) \mid l \in lhs(R) \,\} \cup \{\, sub(r) \mid r \in lhs(R) \,\}$.

The reduction relation $\to_R \subseteq T_\Sigma(X) \times T_\Sigma(X)$ is defined as follows. For any ground terms $s, t \in T_\Sigma$, $s \to_R t$ if there is a pair $l \approx r$ in E and a context $u \in C_\Sigma(X_1)$ such that $s = u[l]$ and $t = u[r]$. Here we say that R rewrites s to t applying the rule $l \to r$. A GTRS R is *equivalent to* a GTRS E, if $\leftrightarrow_R^* = \leftrightarrow_E^*$ holds.

$IRR(R)$ denotes the set of all ground terms irreducible by R. A GTRS R is reduced if for every rule $u \to v$ in R, u is irreducible with respect to $R - \{\, u \to v \,\}$ and v is irreducible with respect to R. For a reduced GTRS R, $IRR(R) \cap sub(R) = sub(R) - lhs(R)$, and $sub(R) - lhs(R)$ is a set of representatives for $sub(R)/\leftrightarrow_R^*$, see Theorem 3.14 on page 162 in [17].

We say that a GTRS R is confluent, locally confluent, terminating, or convergent, if $\to_R$ has the corresponding property.

We recall the following important result.

Proposition 1. [19] Any reduced GTRS R is convergent.

Proposition 2. For a reduced GTRS R, one can reduce a ground term $t \in T_\Sigma$ to its normal form in linear time of $size(t)$. We traverse the term t in postorder. When visiting a position α, we reduce the subterm t/α of t at α to is normal form $t/\alpha\downarrow_R$.

We say that a GTRS R is equivalent to a GTES E if $\leftrightarrow_R^* = \leftrightarrow_E^*$.

Proposition 3. [19] *For a GTES E one can effectively construct an equivalent reduced GTRS R in $O(n \log n)$ time. Here n is the size of E.*

Proof. We briefly recall Snyder's [19] fast ground completion algorithm. We run a congruence closure algorithm for E over the subterm graph of E [4, 15]. In this way we get the representation of the equivalence relation $\leftrightarrow_E^* \cap (sub(E) \times sub(E))$. We compute a set REP of representatives for $sub(E)/\leftrightarrow_E^*$. Then we construct a reduced GTRS R over Σ as follows. We put the rewrite rule $l \to r$ in R if

- $l = f(p_1, \ldots, p_m)$ for some $f \in \Sigma_m$, $m \geq 0$, and $p_1, \ldots, p_m \in REP$,
- $r \in REP$,
- $l \neq r$ and $l \leftrightarrow_E^* r$.

□

We can decide the word problem of a GTES E applying a congruence closure algorithm [4, 15] for the GTES $E_1 = E \cup \{\, p \approx p, q \approx q \,\}$ and then examine whether

p, q are in the same class of the equivalence relation $\leftrightarrow^*_{E_1} \cap (sub(E_1) \times sub(E_1))$. Assume that we want to solve the word problem of a fixed GTES E for varying terms p, q. Then we compute a convergent GTRS over Σ equivalent to E [8, 14, 16, 19]. We compute $p\downarrow_R$ and $q\downarrow_R$, and compare them. If $p\downarrow_R = q\downarrow_R$, then $p \leftrightarrow^*_E q$. Otherwise, $(p, q) \notin \leftrightarrow^*_E$. By Proposition 2, we can decide the word problem of E in linear time. We can also extend the signature. We introduce constants for the equivalence classes of $\leftrightarrow^*_E \cap (sub(E) \times sub(E))$. Then we can construct in $O(n \log n)$ time a reduced GTRS over the extended signature such that $p\downarrow_R = q\downarrow_R$ if and only if $p \leftrightarrow^*_E q$. By Proposition 2, we can decide the word problem of E in linear time. Finally, assume that we want to solve the word problem of a fixed GTES E for a fixed term p and varying term q. Then we can construct in $O(n \log n)$ time a deterministic tree automaton recognizing the $\leftrightarrow^*_E$-class of p [17].
For other completion algorithms on GTRSs see [5, 16]. For further results on GTRSs see [18]. Proposition 1 and Proposition 3 imply the following well known result.

Proposition 4. [19] *For a GTES E and ground terms p, q, one can decide whether $p \leftrightarrow^*_E q$.*

3 Reachability starting from a term attached to a context

Let R be a reduced GTRS over Σ and $s, t \in IRR(R)$. We say that R reaches t starting from s attached to some context, if there is a $u \in C_\Sigma$ such that $u[s] \rightarrow^*_R t$. Let $RAC(s)$ denote the set of all terms $t \in IRR(R)$ which are reachable by R starting from s attached to some context.

Example 1. Let $\Sigma = \Sigma_0 \cup \Sigma_1$, $\Sigma_0 = \{ 0, 1 \}$, and $\Sigma_2 = \{ f \}$. Let GTRS R consist of the equations $f(0,0) \rightarrow 0$ and $f(0,1) \rightarrow 1$. Clearly R is reduced. Then each element of $IRR(R)$ containing 0 is in $RAC(0)$. For example, $f(f(1,0),1) \in RAC(0)$, because $f(f(1,\diamond),1) \in C_\Sigma$ and

$f(f(1,\diamond),1)[0] = f(f(1,0),1) \rightarrow^*_R f(f(1,0),1)$.

Furthermore, $1 \in RAC(0)$, because

$f(\diamond,1)[0] = f(0,1) \rightarrow_R 1$.

Thus each element of $IRR(R)$ containing 1 is in $RAC(0)$. Consequently, $IRR(R) = RAC(0)$.

Lemma 1. *Let R be a reduced GTRS over Σ. For any $s \in sub(R) - lhs(R)$, we can effectively compute $RAC(s) \cap (sub(R) - lhs(R))$.*

Proof. Let $RAC_0 = \{ s \}$. For each $i \geq 0$, let RAC_{i+1} consists of all elements t, where

- $t \in RAC_i$ or
- $t \in sub(R) - lhs(R)$ and there is a rule $f(t_1, \ldots, t_m) \rightarrow t$ in R for some $f \in \Sigma_m$, $t_1, \ldots, t_m \in sub(R) - lhs(R)$, such that $t_j \in RAC_i$ for some $1 \leq j \leq m$, or

• $t \in sub(R) - lhs(R)$ and $t = f(t_1, \ldots, t_m)$ for some $f \in \Sigma_m$, $t_1, \ldots, t_m \in sub(R) - lhs(R)$, and $t_j \in RAC_i$ for some $1 \leq j \leq m$. Then

$$RAC_i \subseteq RAC_{i+1} \subseteq RAC(s) \cap (sub(R) - lhs(R)) \text{ for } i \geq 0\,. \tag{2}$$

Hence there is an integer $0 \leq \ell \leq card(sub(R) - lhs(R))$ such that $RAC_\ell = RAC_{\ell+1}$. Then

$$RAC_\ell = RAC_{\ell+k} \text{ for } k \geq 1\,. \tag{3}$$

Hence

$$RAC_\ell \subseteq RAC(s) \cap (sub(R) - lhs(R))\,. \tag{4}$$

To show the reverse inclusion, we need the following.

Claim 1. *For any $u \in C_\Sigma$ of height $n \geq 0$ and $t \in sub(R) - lhs(R)$, if $u(s) \to_R^* t$, then $t \in RAC_n$.*

Proof. By induction on n.

□

By (2), (3), and Claim 1, $RAC(s) \cap (sub(R) - lhs(R)) \subseteq RAC_\ell$. By (4),

$$RAC(s) \cap (sub(R) - lhs(R)) = RAC_\ell\,.$$

We compute the sets $RAC_0, RAC_1, \ldots, R_{card(sub(R)-lhs(R))}$. In this way we obtain the integer ℓ and $RAC(s) \cap (sub(R) - lhs(R))$.

□

Lemma 2. *For any reduced GTRS R and $s, t \in IRR(R)$, R reaches t starting from s attached to some context if and only if*

(i) $t = u[s]$ for some $u \in C_\Sigma$ or

(ii) $s \in (sub(R) - lhs(R))$, and there are $u \in C_\Sigma$ and $r \in rhs(R)$ such that $t = u[r]$ and R reaches r starting from s attached to some context.

Proof. ($\Rightarrow$) Assume that R reaches t starting from s attached to some context. Then there is $u \in C_\Sigma$ such that $u[s] \to_R^* t$. If $u[s] = t$, then (i) holds. Otherwise, $u[s] \to_R^+ t$. Hence there are $v_1, v_2, z \in C_\Sigma$ and a rule $l \to r$ in R such that $u[s] = v_1[z[s]] \to_R^* v_1[l] \to_R v_1[r] \to_R^* v_2[r] = t$, where

(a) $u = v_1[z]$,

(b) $z[s] \to_R^* l$,

(c) $l \to r \in R$,

(d) $v_1 \to_R^* v_2$ over the ranked alphabet $\Sigma \cup \diamond$.

Hence $t = v_2[r]$, $v_2 \in C_\Sigma$, $r \in rhs(R)$. By (b), $s \in sub(l)$ or $s \in sub(l_1)$ for some $l_1 \in LHS(R)$. Recall that $s \in IRR(R)$. Hence $s \in (sub(R) - lhs(R))$.

($\Leftarrow$) If (i) holds, then R reaches t starting from s attached to some context.

Assume that (ii) holds. Then there is $z \in C_\Sigma$ such that $z[s] \to_R^* r$. Consequently $(u[z])[s] = u[z[s]] \to_R^* u[r] = t$. Hence R reaches t starting from s attached to some context.

□

Lemma 1 and Lemma 2 imply the following result.

Proposition 5. *For any $s, t \in IRR(R)$, we can decide whether R reaches t starting from s attached to some context.*

4 Two trivial semi-decision procedures

We present the well known trivial semi-decision procedure *PRO1* for the ground word problem of variable preserving TESs. We give examples when *PRO1* is efficient. Then we present the trivial semi-decision procedure *PRO2* for the ground word problem of TESs. Note that *PRO2* is a straightforward generalization of *PRO1*.

Procedure *PRO1* Input: A variable preserving TES E over the ranked alphabet Σ and ground terms $p, q \in T_\Sigma$.
Output: 'yes' if $p \leftrightarrow_E^* q$, 'no' or undefined otherwise.
Let $U_0 = \{ p \}$, $V_0 = \{ q \}$, $i = 0$.
repeat
  $i := i + 1$;
  $U_i := U_{i-1} \cup \{ s |$ there is $u \in U_{i-1}$ such that $u \leftrightarrow_E s \}$;
  $V_i := V_{i-1} \cup \{ s |$ there is $u \in V_{i-1}$ such that $u \leftrightarrow_E s \}$;
until ($U_i = U_{i-1}$ or $V_i = V_{i-1}$) or $U_i \cap V_i$ is not empty;
  if $U_i \cap V_i$ is not empty
  then begin output 'yes'; halt end;
output 'no';
halt

For any variable preserving TES E and ground term u, the set $\{ s | u \leftrightarrow_E s \}$ is finite and then effectively computable. Thus for every $i \geq 0$, U_i and V_i, are finite and can be computed effectively. Hence the above procedure can be implemented. Clearly, *PRO1* outputs 'yes' and halts if and only if $p \leftrightarrow_E^* q$. If *PRO1* outputs 'no' and halts, then $(p, q) \notin \leftrightarrow_E^*$.

We adopt the following example of Lynch [13].

Example 2. Let $\Sigma = \Sigma_0 \cup \Sigma_1$, $\Sigma_0 = \{ \$, \# \}$, $\Sigma_1 = \{ f, g \}$. Consider the TES $E = \{ fffx \approx gfx \}$. We raise the problem whether $\$ \leftrightarrow_E^* \#$. On the one hand, the basic Knuth-Bendix completion procedure runs forever on this example [13]. On the other hand, the goal-directed completion procedure outputs 'no' and halts [13]. It is still open whether the goal-directed completion procedure halts on the TES E and any goal [13].

Observe that for each $u \in T_\Sigma$, the set $\{ s \mid u \leftrightarrow_E^* s \}$ is finite. Hence for any $p, q \in T_\Sigma$, *PRO1* outputs the correct answer and halts. For this example, *PRO1* is more efficient than the basic Knuth-Bendix completion procedure, and is at least as efficient as the goal-directed completion procedure [13, 14].

Example 3. Let $\Sigma = \Sigma_0 \cup \Sigma_2$, $\Sigma_0 = \{ \star, \$, \# \}$, and $\Sigma_2 = \{ f \}$. We define the terms $comb_i \in T_\Sigma(X_i)$, $i \geq 1$, as follows. Let $comb_1 = f(x_1, \star)$, $comb_{i+1} = f(x_1, comb_i[x_2, \ldots, x_{i+1}])$ for $i \geq 1$. For example, $comb_3 = f(x_1, f(x_2, f(x_3, \star)))$. Let $n \geq 1$, $p = comb_{2n}[\#, \ldots, \#]$, and $q = comb_{2n}[\$, \ldots, \$]$. We run procedure *PRO1* on the TES $E = \{ \# \approx \$ \}$ and the ground terms p and q. Then

$$card(U_i) = card(V_i) = \begin{pmatrix} 2n \\ i \end{pmatrix} + \begin{pmatrix} 2n \\ i-1 \end{pmatrix} + \cdots + \begin{pmatrix} 2n \\ 1 \end{pmatrix} \text{ for } i = 1, \ldots n,$$

$U_i \cap V_i = \emptyset$ for $i = 0, 1, \ldots n-1$, and
$comb_{2n}[\#, \ldots, \#, \$, \ldots, \$] \in U_n \cap V_n$.
Hence in the nth step, *PRO1* outputs 'yes' and halts.

Example 4. We present Ceitin's [3, 11] semi-Thue system as a TES. Let $\Sigma = \Sigma_0 \cup \Sigma_1$, $\Sigma_0 = \{\,\$\,\}$, and $\Sigma_1 = \{\,a, b, c, d, e\,\}$. E consists of the equations
$acx_1 \approx cax_1$, $adx_1 \approx dax_1$, $bcx_1 \approx cbx_1$, $bdx_1 \approx dbx_1$,
$ecax_1 \approx cex_1$, $edbx_1 \approx dex_1$,
$cdcax_1 \approx cdcaex_1$, $caaax_1 \approx aaax_1$, $daaax_1 \approx aaax_1$.

Proposition 6. [3, 11] *It is undecidable for an arbitrary given ground term $t \in T_\Sigma$ whether $t \leftrightarrow_E^* a^3\$$.*

We run procedure *PRO1* on the TES E and the ground terms $p = a^3\$$ and $q = edb\$$. We compute as follows.
$U_0 = \{\,p\,\}$, $V_0 = \{\,q\,\}$,
$U_1 = \{\,a^3\$, ca^3\$, da^3\$\,\}$, $V_1 = \{\,edb\$, ebd\$, de\$\,\}$,
$U_2 = \{\,a^3\$, ca^3\$, da^3\$, cca^3\$, cda^3\$, dca^3\$, dda^3\$, acaa\$, adaa\$\,\}$, $V_2 = V_1$.
Now procedure *PRO1* outputs 'no' and halts.

Let $n \geq 1$, $p = (bd)^{2n}\$$, and $q = (db)^{2n}\$$. We apply procedure *PRO1* to TES E and ground terms p and q. We compute as follows.
$U_0 = \{\,p\,\}$, $V_0 = \{\,q\,\}$,
$U_1 = \{\,p, db(bd)^{2n-1}\$, \ldots, (bd)^{2n-1}db\$\,\}$,
$V_1 = \{\,q, bd(db)^{2n-1}\$, \ldots, (db)^{2n-1}bd\$,\,\}$,
$U_2 = U_1 \cup \{\; dbdb(bd)^{2n-2}\$, dbbddb(bd)^{2n-3}\$, \ldots, (bd)^{2n-2}dbdb\$\,\}$,
$V_2 = V_1 \cup \{\,bdbd(db)^{2n-2}\$, bddbbd(db)^{2n-3}\$, \ldots, (db)^{2n-2}bdbd\$\,\}$,
$\ldots$.
Observe that $U_i \cap V_i = \emptyset$ for $i = 0, 1, \ldots, n-1$. Clearly, $(bd)^n(db)^n\$ \in U_n \cap V_n$. After computing U_n and V_n, procedure *PRO1* outputs 'yes' and halts.

Example 5. We continue Example 4. Let $p \in T_\Sigma$ be arbitrary such that symbols a or c appear in p. Let $q \in T_\Sigma$ such that a, c do not appear in q. That is, only the constant \$ and the symbols b, d, or e appear in q.

Observe that the left-hand side and the right-hand side of the fourth and sixth rules do not contain a or c. Both sides of all other rules contain a or c. Hence for any reduction sequence
$p \to_R p_1 \to_R p_2 \to_{\ldots} \to_R p_n$, $n \geq 1$, for any $1 \leq i \leq n$, the term p_i contains the constant \$ and at least one a or c. Furthermore, along any reduction sequence $q \to_R q_1 \to_R q_2 \to_{\ldots} \to_R q_n$, $n \geq 1$, we only use the fourth and sixth equations. Consequently, the set $\{\,v \in T_\Sigma \mid q \leftrightarrow_E^* v\,\}$ is finite. Furthermore neither a nor c appears in any element of the set $\{\,v \in T_\Sigma \mid q \leftrightarrow_E^* v\,\}$. Thus

$$(p, q) \notin \overset{*}{\underset{E}{\leftrightarrow}}, \tag{5}$$

and $U_i \cap V_i = \emptyset$ for $i \geq 0$. Thus procedure *PRO1* outputs 'no' and halts on the input E, p, q.

Example 6. Let $\Sigma = \Sigma_0 \cup \Sigma_1$, $\Sigma_0 = \{ a \}$, and $\Sigma_1 = \{ f \}$. TES E consists of the equation $fffx \approx x$. We run procedure *PRO1* on TES E and ground terms $p = a$ and $q = fa$. We compute as follows.
$U_0 = \{ a \}$, $V_0 = \{ fa \}$,
$U_1 = \{ a, fffa \}$, $V_1 = \{ fa, f^3a \}$,
$U_2 = \{ a, fffa, f^4a \}$, $V_2 = \{ fa, f^3a, f^5a \}$,....
$U_0 \subset U_1 \subset U_2 \subset \cdots$,
$V_0 \subset V_1 \subset V_2 \subset \cdots$, and
$U_i \cap V_i = \emptyset$ for $i \geq 0$.
Hence procedure *PRO1* does not halt.

To present the semi-decision procedure *PRO2*, we define the sets $U_i \subseteq T_\Sigma$, $i \geq 0$, by recursion. Let $U_0 = \{ p \}$. Let $i \geq 1$. We put all elements of U_{i-1} in U_i. Moreover, we put in U_i all $s \in T_\Sigma$ such that

- $l' \approx r'$ is a ground instance of some equation $l \approx r$ in $E \cup E^{-1}$ obtained by substituting arbitrary ground terms of height less than or equal to $i - 1$ for all variables that do not appear in l,
- $v \in C_\Sigma$,
- $v[l'] \in U_{i-1}$ and $s = v[r']$.

We define $V_i \subseteq T_\Sigma$, $i \geq 0$, symmetrically to U_i, $i \geq 0$. Clearly for every $i \geq 0$, U_i and V_i are finite and can be computed effectively. Note that there may be an $i \geq 1$ such that $U_i = U_{i+1}$ and $U_{i+1} \subset U_{i+2}$.

Example 7. Let $\Sigma = \Sigma_0 \cup \Sigma_1$, $\Sigma_0 = \{ 0, 1 \}$, and $\Sigma_2 = \{ f \}$. Let TES E consist of the equations

$f(x_1, x_1) \approx 0$, $f(0, x_1) \approx x_1$.

Let $p = f(1, 0)$ and $q = f(1, f(1, 1))$. Then
$U_0 = \{ f(1, 0) \}$, $V_0 = \{ f(1, f(1, 1)) \}$,
$U_1 = \{ f(1, 0), f(f(0, 1), 0), f(1, f(0, 0)), f(1, f(1, 1)) \}$,
$V_1 = \{ f(1, f(1, 1)), f(f(0, 1), f(1, 1)), f(1, 0), f(1, f(f(0, 1), 1)),$
$f(1, f(1, f(0, 1))) \}$.

Procedure *PRO2* Input: A TES E over the ranked alphabet Σ and ground terms $p, q \in T_\Sigma$.
Output: 'yes' if $p \leftrightarrow_E^* q$, undefined otherwise.

```
1   i := i + 1;
    compute U_i and V_i;
    if U_i ∩ V_i is not empty then begin output 'yes'; halt end;
    goto 1
```

PRO2 outputs 'yes' and halts if and only if $p \leftrightarrow_E^* q$.

Example 8. We continue Example 7. We run procedure *PRO2* on TES E and ground terms p, q. We compute as follows. We compute U_0 and V_0. We observe that $U_0 \cap V_0$ is empty. Then we compute U_1 and V_1. We observe that $U_1 \cap V_1$ is not empty. Procedure *PRO2* outputs 'yes' and halts.

5 Semi-decision procedure for the ground word problem of variable preserving TESs

We present the semi-decision procedure *PRO3* for the ground word problem of variable preserving TESs, and show its correctness. *PRO3* is an improvement of *PRO1*. The starting idea is the following. For each $i \geq 1$, we construct the GTES P_i using those instances of equations in $E \cup E^{-1}$ which are applied to compute the set U_i. We improve this construction by defining P_i, $i \geq 2$, as the set of all instances of equations in $E \cup E^{-1}$ which can be applied to elements of $\{ s \in T_\Sigma \mid p \leftrightarrow^*_{P_{i-1}} s \}$ rather than to the elements of U_{i-1}. Furthermore, we define the GTES Q_i symmetrically. We give examples when procedure *PRO3* is more efficient than procedure *PRO1*.

Let E be a variable preserving TES over Σ, and let $p, q \in T_\Sigma$. We define the GTESs P_i and the reduced GTRSs R_i, $i \geq 1$, over Σ as follows.

For each equation $l \approx r$ of $E \cup E^{-1}$ with $l, r \in T_\Sigma(X_m)$, $m \geq 0$, and for any $u \in C_\Sigma$, $u_1, \ldots, u_m \in T_\Sigma$, if $p = u[l[u_1, \ldots, u_m]]$ then we put the equation $l[u_1, \ldots, u_m] \approx r[u_1, \ldots, u_m]$ in P_1. Applying Snyder's algorithm we compute a reduced GTRS R_1 equivalent to the GTES P_1, see Proposition 3.

Let $i \geq 1$. (a) We put each element of R_i into P_{i+1}.

(b) For each equation $l \approx r$ of $E \cup E^{-1}$, $l, r \in T_\Sigma(X_m)$, $m \geq 0$, for any $u_1, \ldots, u_m \in (sub(R_i) - lhs(R_i)) \cup sub(p{\downarrow_{R_i}})$, if R_i reaches $p{\downarrow_{R_i}}$ starting from $l[u_1, \ldots, u_m]{\downarrow_{R_i}}$ attached to some context, and $l[u_1, \ldots, u_m]{\downarrow_{R_i}} \neq r[u_1, \ldots, u_m]{\downarrow_{R_i}}$, then we put the equation $l[u_1, \ldots, u_m] \approx r[u_1, \ldots, u_m]$ in P_{i+1}.

If $P_{i+1} = R_i$, then let $R_{i+1} = R_i$. Otherwise, applying Snyder's algorithm, we compute a reduced GTRS R_{i+1} equivalent to the GTES P_{i+1}.

When misunderstanding may arise, we denote R_i as R_{P_i}. We define the GTESs Q_i, $i \geq 1$, symmetrically to the GTESs P_i, $i \geq 1$. Applying Snyder's algorithm, we compute a reduced GTRS $R_{P_i \cup Q_i}$ equivalent to the GTRS $R_{P_i} \cup R_{Q_i}$ for $i \geq 1$.

We illustrate our concepts and results by the following example.

Example 9. Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$, $\Sigma_0 = \{ \$, \# \}$, $\Sigma_1 = \{ e, f, g, h \}$, and $\Sigma_2 = \{ d \}$. Let the TES E consist of the equations

$\# \approx \$$, $g\$ \approx h\$$, $d(hx_1, hx_1) \approx hx_1$, $efhx_1 \approx hx_1$.

Observe that E is variable preserving. Let $p = efg\#$, $q = d(h\#, h\#)$.

First we compute the GTES P_i, $i \geq 1$. GTES P_1 consists of the equation $\# \approx \$$. Let Θ stand for $\leftrightarrow^*_{P_1} \cap (sub(P_1) \times sub(P_1))$. Then $sub(P_1)/\Theta = \{ \{ \#, \$ \} \}$ and $\{ \$ \}$ is a set of representatives for $sub(P_1)/\leftrightarrow^*_{P_1}$. GTRS R_1 consists of the rule $\# \to \$$.

GTES P_2 consists of the equations $\# \approx \$$, $g\$ \approx h\$$. Let Θ stand for $\leftrightarrow^*_{P_2} \cap (sub(P_2) \times sub(P_2))$. Then $sub(P_2)/\Theta = \{ \{ \#, \$ \}, \{ g\#, g\$, h\#, h\$ \} \}$ and $\{ \$, h\$ \}$ is a set of representatives for $sub(P_2)/\leftrightarrow^*_{P_2}$. GTRS R_2 consists of the rules $\# \to \$$, $g\$ \to h\$$.

GTES P_3 consists of the equations

$\# \approx \$$, $g\$ \approx h\$$, $h\$ \approx d(h\$, h\$)$, $h\$ \approx efh\$$.

Let Θ stand for $\leftrightarrow^*_{P_3} \cap (sub(P_3) \times sub(P_3))$. Then

$sub(P_3)/\Theta = \{\{\#, \$\}, \{g\#, g\$, h\#, h\$, d(h\$, h\$), efh\$\}, \{fh\$\}\}$
and $\{\$, h\$, fh\$\}$ is a set of representatives for $sub(P_3)/\leftrightarrow^*_{P_3}$. R_3 consists of the rules

$\# \to \$, \quad g\$ \to h\$, \quad d(h\$, h\$) \to h\$, \quad efh\$ \to h\$.$

$P_4 = R_3$ and $R_4 = R_3$. Furthermore, $P_i = R_3$ and $R_i = R_3$ for $i \geq 4$.

Second, we compute the GTESs Q_i, $i \geq 1$. GTES Q_1 consists of the equations $\# \approx \$$, $d(h\#, h\#) \approx h\#$. GTRS R_{Q_1} consists of the rules $\# \to \$$, $d(h\$, h\$) \to h\$$.

GTES Q_2 consists of the equations $\# \approx \$$, $d(h\$, h\$) \approx h\$$, $efh\$ \approx h\$$.

GTRS R_{Q_2} consists of the rules $\# \to \$$, $d(h\$, h\$) \to h\$$, $efh\$ \to h\$$.

Observe that $R_{Q_2} = Q_i = R_{Q_i}$ for $i \geq 3$.

$R_{P_1 \cup Q_1} = R_{P_1}$, $R_{P_2 \cup Q_2} = R_{P_2} \cup R_{Q_2}$, and $R_{P_3 \cup Q_3} = R_{P_3}$. Then

$p{\downarrow_{R_{P_1 \cup Q_1}}} = efg\$$, $q{\downarrow_{R_{P_1 \cup Q_1}}} = h\$$,

$p{\downarrow_{R_{P_2 \cup Q_2}}} = h\$$, $q{\downarrow_{R_{P_2 \cup Q_2}}} = h\$$.

We get the following result by direct inspection of the definition of the GTES P_i, $i \geq 1$.

Lemma 3. *(a) For each $i \geq 1$, $\leftrightarrow^*_{P_i} = \leftrightarrow^*_{R_i} \subseteq \leftrightarrow^*_{P_{i+1}} \subseteq \leftrightarrow^*_E$.*

(b) If $R_i = P_{i+1}$ for some $i \geq 1$, then $R_i = P_j = R_j$ for $j \geq i + 1$.

Lemma 4. *For each $i \geq 1$, we can effectively construct the GTES P_i.*

Proof. By induction on i.

Base Case: $i = 1$. Clearly, we can construct P_1.

Induction Step: Let $i \geq 1$. Assume that we have constructed P_i. By Proposition 3, we can construct R_i. Consider item (b) in the definition of P_i. By Proposition 5, we can effectively decide whether R_i reaches $p{\downarrow_{R_i}}$ starting from $l[u_1 \ldots, u_m]{\downarrow_{R_i}}$ attached to some context. Hence we can construct P_{i+1} as well.

□

We now present our semi-decision procedure.

Procedure *PRO3 Input:* A variable preserving TES E over the ranked alphabet Σ and ground terms $p, q \in T_\Sigma$.

Output:
- 'yes' if $p \leftrightarrow^*_E q$,
- 'no' if $(p, q) \notin \leftrightarrow^*_E$ and the procedure halts,
- undefined if the procedure does not halt.

compute P_1, R_{P_1}, Q_1, R_{Q_1}, and $R_{P_1 \cup Q_1}$;
if $p{\downarrow_{R_{P_1 \cup Q_1}}} = q{\downarrow_{R_{P_1 \cup Q_1}}}$, then begin output 'yes'; halt end;
$i := 1$;
1: $i := i + 1$;
compute P_i, R_{P_i}, Q_i, R_{Q_i}, and $R_{P_i \cup Q_i}$;
if $p{\downarrow_{R_{P_i \cup Q_i}}} = q{\downarrow_{R_{P_i \cup Q_i}}}$, then begin output 'yes'; halt end;
if $R_{P_{i-1}} = P_i$ or $R_{Q_{i-1}} = Q_i$,
then begin output 'no'; halt end;
goto 1

Example 10. We continue Example 9. Note that $p{\downarrow_{R_{P_1\cup Q_1}}} \neq q{\downarrow_{R_{P_1\cup Q_1}}}$. Hence procedure *PRO3* does not output anything and does not halt in the first step. Observe that $p{\downarrow_{R_{P_2\cup Q_2}}} = q{\downarrow_{R_{P_2\cup Q_2}}}$. Hence procedure *PRO3* outputs 'yes' and halts in the second step.

Example 11. We continue Example 5. Let $n \geq 1$. We run procedure *PRO3* on the TES E and the ground terms $p = (bd)^{2n}\$$, and $q = (db)^{2n}\$$. We compute as follows. GTES P_1 consists of the equation $bd\$ \approx db\$$. Let Θ stand for $\leftrightarrow^*_{P_1} \cap (sub(P_1) \times sub(P_1))$. Then $sub(P_1)/\Theta = \{ \{ b\$ \}, \{ d\$ \}, \{ bd\$ \} \}$ and $\{ bd\$ \}$ is a set of representatives for $sub(P_1)/\leftrightarrow^*_{P_1}$. GTRS R_{P_1} consists of the rule $bd\$ \to db\$$.

Symmetrically, GTES Q_1 consists of the equation $db\$ \approx bd\$$. GTRS R_{Q_1} consists of the rule $db\$ \to bd\$$. It is not hard to see, that GTRS $R_{P_1\cup Q_1}$ is equal to GTRS R_{P_1}. Observe that $p{\downarrow_{R_{P_1\cup Q_1}}} = q{\downarrow_{R_{P_1\cup Q_1}}}$, Hence procedure *PRO3* outputs 'yes' and halts in the first step.

We run procedure *PRO3* on the TES E and the ground terms $p = aaa\$$ and $q = bedb\$$. By our arguments in Example 5,

$$p{\downarrow_{R_{P_i\cup Q_i}}} \neq q{\downarrow_{R_{P_i\cup Q_i}}} \quad \text{for } i \geq 1\,.$$

Furthermore, *PRO3* computes as follows.

$R_{Q_1} = \{ db\$ \to bd\$, edb\$ \to de\$ \}$,

$R_{Q_2} = \{ db\$ \to bd\$, edb\$ \to de\$, bdde\$ \to dbde\$ \}$, and

$R_{Q_2} = R_{Q_{n+2}}$ for $n \geq 1$.

Consequently, Procedure *PRO3* outputs 'no' and then halts. Generalizing our arguments, we can show the following.

Statement 1. *Let $p \in T_\Sigma$ be arbitrary such that symbols a or c appear in p. Let $q \in T_\Sigma$ such that a, c do not appear in q. Then procedure PRO3 outputs 'no' and halts on the input E, p, q.*

By Propositon 6, for an arbitrary ground term $q' \in T_\Sigma$, the goal-directed completion procedure [13] may fail or may not halt on the TES E and the goal $(aaa\$, q')$. The following problem is open. For each goal $(aaa\$, q)$ such that $q \in T_\Sigma$, and a, c do not appear in q, is it true that the the goal-directed completion procedure does not fail and halts on the TES E and the goal $(aaa\$, q)$.

It is open whether the goal-directed completion procedure does not fail and halts on the TES E and any goal $(aaa\$, q)$ such that $q \in T_\Sigma$, a, c do not appear in q.

We now show the correctness of Procedure *PRO3*.

Lemma 5. *For any i, n with $1 \leq n \leq i$, and any $t_1, \ldots, t_n \in T_\Sigma$, if $p \leftrightarrow_E t_1 \leftrightarrow_E t_2 \leftrightarrow_E \cdots \leftrightarrow_E t_n$, then $p \leftrightarrow^*_{P_i} t_1 \leftrightarrow^*_{P_i} t_2 \leftrightarrow^*_{P_i} \cdots \leftrightarrow^*_{P_i} t_n$.*

Proof. We proceed by induction on i.

Base Case: $i = 1$. Then $n = 1$. By the definition of P_1, we have $p \leftrightarrow_{P_1} t_1$.

Induction Step: Let $i \geq 1$, and assume that the statement holds for $1, 2, \ldots, i$. We now show that the statement holds for $i+1$. To this end, assume that

$$p \underset{E}{\leftrightarrow} t_1 \underset{E}{\leftrightarrow} t_2 \underset{E}{\leftrightarrow} \cdots \underset{E}{\leftrightarrow} t_n \text{ for some } 0 \leq n \leq i+1\,. \tag{6}$$

By the induction hypothesis,

$$p \overset{*}{\underset{P_i}{\leftrightarrow}} t_1 \overset{*}{\underset{P_i}{\leftrightarrow}} t_2 \overset{*}{\underset{P_i}{\leftrightarrow}} \cdots \overset{*}{\underset{P_i}{\leftrightarrow}} t_{n-1}\,. \tag{7}$$

Hence

$$t_{n-1} \overset{*}{\underset{R_i}{\rightarrow}} p{\downarrow_{R_i}}\ . \tag{8}$$

By (6), there is an equation $l \approx r$ in $E \cup E^{-1}$ with $l, r \in T_\Sigma(X_m)$, $m \geq 0$ and there are $u \in C_\Sigma$, $u_1, \ldots, u_m \in T_\Sigma$ such that

$$t_{n-1} = u[l[u_1, \ldots, u_m]] \text{ and } t_n = u[r[u_1, \ldots, u_m]]\,. \tag{9}$$

As R_i is convergent, by (8) and (9), $u[l[u_1, \ldots, u_m]{\downarrow_{R_i}}] \rightarrow_{R_i}^* p{\downarrow_{R_i}}$. That is, R_i reaches $p{\downarrow_{R_i}}$ starting from $l[u_1, \ldots, u_m]{\downarrow_{R_i}}$ attached to some context. By the definition of P_{i+1},

$$l[u_1, \ldots, u_m] \approx r[u_1, \ldots, u_m] \text{ is in } \overset{*}{\underset{P_i}{\leftrightarrow}} \text{ or } P_{i+1}\,. \tag{10}$$

By Lemma 3, (7), (9), and (10),

$$p \overset{*}{\underset{P_{i+1}}{\leftrightarrow}} t_1 \overset{*}{\underset{P_{i+1}}{\leftrightarrow}} t_2 \overset{*}{\underset{P_{i+1}}{\leftrightarrow}} \cdots \overset{*}{\underset{P_{i+1}}{\leftrightarrow}} t_{n-1} \overset{*}{\underset{P_{i+1}}{\leftrightarrow}} t_n\,.$$

□

By Lemma 3 and Lemma 5 we have the following result.

Lemma 6. *Assume that $R_i = P_{i+1}$ for some $i \geq 1$. Then $p \leftrightarrow_{P_{i+1}}^* q$ if and only if $p \leftrightarrow_E^* q$.*

Theorem 1. *If $p \leftrightarrow_E^* q$, then procedure PRO3 outputs 'yes' and halts.*

Proof. Assume that $p = t_1 \leftrightarrow_E t_2 \leftrightarrow_E \cdots \leftrightarrow_E t_n = q$ for some $n \geq 1$ and $t_1, \ldots, t_n \in T_\Sigma$. By Lemma 5, $p \leftrightarrow_{P_n}^* q$. Let k be the least integer such that $p \leftrightarrow_{P_k \cup Q_k}^* q$.

First assume that $k = 1$. Then $p \leftrightarrow_{P_1 \cup Q_1}^* q$. Hence $p{\downarrow_{R_{P_1 \cup Q_1}}} = q{\downarrow_{R_{P_1 \cup Q_1}}}$. Consequently, procedure *PRO3* outputs 'yes' and halts in the first step.

Second assume that $k \geq 2$. Then by the definition of k, $(p, q) \notin \leftrightarrow_{P_i \cup Q_i}^*$ for $2 \leq i \leq k-1$. Then by Lemma 6, $R_{P_{i-1}} \subset P_i$ and $R_{Q_{i-1}} \subset Q_i$ for $2 \leq i \leq k-1$. Hence procedure *PRO3* does not halt in the first $k-1$ steps. By the definition of the integer k, in the kth step procedure *PRO3* outputs 'yes' and halts.

□

Theorem 2. *If procedure PRO3 outputs 'yes' and halts, then* $p \leftrightarrow_E^* q$. *If procedure PRO3 outputs 'no' and halts, then* $(p,q) \notin \leftrightarrow_E^*$.

Proof. Assume that procedure *PRO3* outputs 'yes' and halts in the kth step. Then $p \leftrightarrow_{P_k \cup Q_k}^* q$. By Lemma 3, $p \leftrightarrow_E^* q$.

Assume that procedure *PRO3* outputs 'no' and halts in the kth step. Then

(a) $(p,q) \notin \leftrightarrow_{P_k \cup Q_k}^*$ and

(b) $P_k = R_{P_{k-1}}$ or $Q_k = R_{Q_{k-1}}$.

We now distinguish two cases.

Case 1: $P_k = R_{P_{k-1}}$. By (a) and by Lemma 6, $(p,q) \notin \leftrightarrow_E^*$.

Case 2: $Q_k = R_{Q_{k-1}}$. This case is symmetric to Case 2.

□

Theorems 1 and 2 imply the following.

Theorem 3. *If* $p \leftrightarrow_E^* q$, *then procedure PRO3 outputs 'yes' and halts. Otherwise, either PRO3 outputs 'no' and halts, or PRO3 does not halt.*

Example 12. We continue Example 3. We now run procedure *PRO3* on the TES E and the ground terms p, q. Then $P_1 = Q_1 = \{ \# \approx \$ \}$, $R_{P_1} = R_{Q_1} = P_1$, and $R_{P_1 \cup Q_1} = P_1$. Observe that $p{\downarrow_{R_{P_1 \cup Q_1}}} = q{\downarrow_{R_{P_1 \cup Q_1}}}$. Hence procedure *PRO3* outputs 'yes' and halts in the first step. By Proposition 2, we compute $p{\downarrow_{R_{P_1 \cup Q_1}}}$ and $q{\downarrow_{R_{P_1 \cup Q_1}}}$ in linear time. We apply the rules of $R_{P_1 \cup Q_1}$ n times. For this example, *PRO3* is faster than *PRO1*.

Example 13. We continue Example 6. We now run procedure *PRO3* on the TES E and the ground terms p and q. Then $\{ a \approx fffa \} = P_1 = R_{P_1} = P_{1+i} = R_{P_{1+i}}$ for $i \geq 1$. Furthermore, $Q_1 = \{ a \approx fffa,\ fa \approx ffffa \}$, $R_{Q_1} = P_1 = Q_2 = R_{Q_2} = Q_{1+i} = R_{Q_{1+i}}$ for $i \geq 1$.

Observe that $p{\downarrow_{R_{P_2 \cup Q_2}}} \neq q{\downarrow_{R_{P_2 \cup Q_2}}}$. Hence procedure *PRO3* outputs 'no' and halts in the second step.

It should be clear that for all ground terms p and q, *PRO3* halts. It outputs 'yes' if $p \leftrightarrow_E^* q$. Otherwise it outputs 'no'.

Example 14. We now continue Example 2. We apply procedure *PRO3* to the TES $E = \{ fffx \approx gfx \}$ and any terms $p, q \in T_\Sigma$. Observe that $height(fffx) = 2 = height(gfx)$.

Statement 2. *For each* $i \geq 0$, *and for each pair of terms,* $s, t \in T_\Sigma(X)$, *if* $(s,t) \in P_i$, *then* $height(s) = height(t) \leq height(p)$.

Proof. We proceed by induction on n.

Base Case: $i = 1$. By the definition of P_1, for each equation $s \approx t$ in P_1, $height(s) = height(t) \leq height(p)$. Hence our statement holds.

Induction Step: Let $n \geq 1$, and assume that the satement holds for $1, 2, \ldots, n$. We now show that the satement holds for $n+1$. Consider an equation

$l[u_1, \ldots, u_m] \approx r[u_1, \ldots, u_m]$ in P_{i+1}. Then there exist

• an equation $l \approx r$ of $E \cup E^{-1}$, where $l, r \in T_\Sigma(X_m)$, $m \geq 0$.

• $u_1, \ldots, u_m \in (sub(R_i) - lhs(R_i)) \cup sub(p{\downarrow_{R_i}})$.

such that R_i reaches $p{\downarrow_{R_i}}$ starting from $l[u_1, \ldots, u_m]{\downarrow_{R_i}}$ attached to some context, and that

$l[u_1, \ldots, u_m]{\downarrow_{R_i}} \neq r[u_1, \ldots, u_m]{\downarrow_{R_i}}$.

Consequently, there is a $u \in C_\Sigma$ such that $u[l[u_1, \ldots, u_m]] \rightarrow^*_{R_i} p$. By (a) in Lemma 3 and the induction hypothesis, $height(u[l[u_1, \ldots, u_m]]) = height(p)$. Thus $height(l[u_1, \ldots, u_m]) \leq height(p)$. By (a) in Lemma 3 and the induction hypothesis, $height(l) = height(r)$. Hence $height(l[u_1, \ldots, u_m]) = height(r[u_1, \ldots, u_m])$. □

Observe that the set $\{ (s,t) \in T_\Sigma \times T_\Sigma \mid height(s) = height(t) \leq height(p) \}$ is finite. By Lemma 3 and Statement 2, procedure *PRO3* halts on E and any terms $p, q \in T_\Sigma$ in finitely many steps.

The following result can be shown by generalizing the proof appearing in Example 14.

Theorem 4. *Let E be a variable preserving TES such that*

• *for any equation $s \approx t$ in E, $height(s) = height(t)$, or*

• *for any equation $s \approx t$ in E, $size(s) = size(t)$ and each variable appears the same times in s and t.*

Let $p, q \in T_\Sigma$ be arbitrary. Then procedure PRO3 halts on E and terms p, q.

6 Semi-decision procedure for the ground word problem of TESs

We present the semi-decision procedure *PRO4* for the ground word problem of TESs, and show its correctness. We obtain it generalizing *PRO3* taking into account *PRO2*. The starting point to the definition of the GTESs P_i, $i \geq 1$, is the same as in Section 5. We define P_1 as the set of all instances $l' \rightarrow r'$ of equations $l \approx r$ in $E \cup E^{-1}$ which can be applied to p. We define P_{i+1}, $i \geq 1$, as the set of all instances $l' \rightarrow r'$ of equations $l \approx r$ in $E \cup E^{-1}$ which can be applied to elements of $\{ s \in T_\Sigma \mid p \leftrightarrow^*_{P_i} s \}$. The question is what should we substitute for those variables in the right-hand side r that do not appear in the left-hand side l. We now give a simplified answer to this question. Applying Snyder's algorithm we compute a reduced GTRS R_i equivalent to the GTES P_i. When constructing the instance $l' \rightarrow r'$ of $l \approx r$, we substitute any term in $(sub(R_i) - lhs(R_i)) \cup sub(p{\downarrow_{R_i}})$ or the R_i normal form of any ground term of height less than or equal to i for each variable in the right-hand side r that does not appear on the left-hand side l. Furthermore, we define the GTESs Q_i, $i \geq 1$, symmetrically.

Let E be a TES over Σ, and let $p, q \in T_\Sigma$. We now define the GTESs P_i and the reduced GTRSs R_i, $i \geq 1$, over Σ.

Let $NORM_0 = \Sigma_0 \cup sub(p)$. For each equation $l \approx r$ of $E \cup E^{-1}$ with $l \in T_\Sigma(X_{k+m})$, $r \in T_\Sigma(X_k \cup X_{[k+m+1,k+m+\ell]})$ for some $k, m, \ell \geq 0$, if $p = u[l[u_1, \ldots, u_{k+m}]]$ for some $u \in C_\Sigma$, $u_1, \ldots, u_{k+m} \in T_\Sigma$, then for all

$v_{k+m+1}, \ldots, v_{k+m+\ell} \in NORM_0$, we put the equation

$$l[u_1, \ldots, u_{k+m}] \approx r[u_1, \ldots, u_k, v_{k+m+1}, \ldots, v_{k+m+\ell}]$$

in P_1. Applying Snyder's algorithm we compute a reduced GTRS R_1 equivalent to the GTES P_1, see Proposition 3.

Let $i \geq 1$. Let

$NORM_i = sub(p{\downarrow_{R_i}}) \cup (sub(R_i) - lhs(R_i)) \cup$

$\{ t{\downarrow_{R_i}} \mid t \in NORM_{i-1}$ or $t = f(t_1, \ldots, t_m)$ for some $f \in \Sigma_m$ and $t_1, \ldots, t_m \in NORM_{i-1} \}$.

(a) We put each rule of R_i into P_{i+1}.

(b) For each equation $l \approx r$ of $E \cup E^{-1}$ with $l \in T_\Sigma(X_{k+m})$, $r \in T_\Sigma(X_k \cup X_{[k+m+1,k+m+\ell]})$ for some $k, m, \ell \geq 0$, for any $u_1, \ldots, u_{k+m} \in (sub(R_i) - lhs(R_i)) \cup sub(p{\downarrow_{R_i}})$ and $v_{k+m+1}, \ldots, v_{k+m+\ell} \in NORM_i$, if R_i reaches $p{\downarrow_{R_i}}$ starting from $l[u_1, \ldots, u_{k+m}]{\downarrow_{R_i}}$ attached to some context, and

$l[u_1, \ldots, u_{k+m}]{\downarrow_{R_i}} \neq r[u_1, \ldots, u_m, v_{k+m+1}, \ldots, v_{k+m+\ell}]{\downarrow_{R_i}}$,

then we put the equation

$$l[u_1, \ldots, u_{k+m}] \approx r[u_1, \ldots, u_m, v_{k+m+1}, \ldots, v_{k+m+\ell}]$$

in P_{i+1}.

If we do not put equations in P_{i+1} in item (b), i.e. $P_{i+1} = R_i$, then let $R_{i+1} = R_i$. Otherwise, applying Snyder's algorithm, we compute a reduced GTRS R_{i+1} equivalent to the GTES P_{i+1}.

When misunderstanding may arise, we denote R_i as R_{P_i}. We define the GTESs Q_i, $i \geq 1$, symmetrically to the GTESs P_i, $i \geq 1$. Applying Snyder's algorithm, we compute a reduced GTRS $R_{P_i \cup Q_i}$ equivalent to the GTRS $R_{P_i} \cup R_{Q_i}$ for $i \geq 1$.

By Proposition 1 GTRSs R_{P_i}, R_{Q_i}, and $R_{P_i \cup Q_i}$ are convergent.

We illustrate our concepts and results by two running examples, each of them is presented as a series of examples.

Example 15. We continue Example 7. Let $p = f(0,1)$ and $q = f(f(0,1),1)$. Observe that for any $u, v \in T_\Sigma$, if $u \overset{*}{\underset{E}{\leftrightarrow}} v$, then the parity of the number of 1's in u equals to that in v. Hence

$$(p, q) \notin \overset{*}{\underset{E}{\leftrightarrow}} . \tag{11}$$

We now construct the GTESs P_1, P_2, and P_3. Then $NORM_0 = \{ 0, 1, f(0,1) \}$. P_1 consists of the equations

$0 \approx f(0,0)$, $0 \approx f(1,1)$, $0 \approx f(f(0,1), f(0,1))$,

$1 \approx f(0,1)$, $f(0,1) \approx 1$, $f(0,1) \approx f(0, f(0,1))$.

R_1 consists of the rules

$f(0,0) \to 0$, $f(1,1) \to 0$, $f(0,1) \to 1$.

$NORM_1 = \{ 0, 1, f(1,0) \}$. P_2 consists of the equations

$f(0,0) \approx 0$, $f(1,1) \approx 0$, $f(0,1) \approx 1$, $0 \approx f(f(1,0), f(1,0))$.

R_2 consists of the rules

$f(0,0) \to 0$, $f(1,1) \to 0$, $f(0,1) \to 1$, $f(f(1,0), f(1,0)) \to 0$.

$NORM_2 = \{\, 0,\ 1,\ f(1,0),\ f(0,f(1,0)),\ f(1,f(1,0)),\ f(f(1,0),0),\ f(f(1,0),1)\,\}$.
P_3 consists of the equations
$f(0,0) \approx 0, \quad f(1,1) \approx 0, \quad f(0,1) \approx 1, \quad f(f(1,0),f(1,0)) \approx 0,$
$0 \approx f(f(0,f(1,0)),f(0,f(1,0)))$,
$0 \approx f(f(1,f(1,0)),f(1,f(1,0)))$,
$0 \approx f(f(f(1,0),0),f(f(1,0),0))$,
$0 \approx f(f(f(1,0),1),f(f(1,0),1))$.
R_3 consists of the rules
$f(0,0) \to 0, \quad f(1,1) \to 0, \quad f(0,1) \to 1, \quad f(f(1,0),f(1,0)) \to 0,$
$f(f(0,f(1,0)),f(0,f(1,0))) \to 0$,
$f(f(1,f(1,0)),f(1,f(1,0))) \to 0$,
$f(f(f(1,0),0),f(f(1,0),0)) \to 0$,
$f(f(f(1,0),1),f(f(1,0),1)) \to 0$.
Continuing in this manner we get that

$$R_{P_i} \subset R_{P_{i+1}} \text{ for } i \geq 1\,. \tag{12}$$

We now compute the GTESs Q_1, Q_2, and Q_3.
$NORM_0 = \{\, 0, 1, f(0,1), f(f(0,1),1)\,\}$.
Q_1 consists of the equations
$0 \approx f(0,0), \quad 0 \approx f(1,1), \quad 0 \approx f(f(0,1),f(0,1)),$
$0 \approx f(f(f(0,1),1),f(f(0,1),1))$,
$1 \approx f(0,1), \quad f(0,1) \approx f(0,f(0,1)), \quad f(f(0,1),1) \approx f(0,f(f(0,1),1))$.
R_{Q_1} consists of the rules
$f(0,0) \to 0, \quad f(1,1) \to 0, \quad f(0,1) \to 1$.
$NORM_1 = \{\, 0, 1, f(1,0)\,\}$.
Q_2 consists of the equations
$f(0,0) \approx 0, \quad f(1,1) \approx 0, \quad f(0,1) \approx 1, \quad 0 \approx f(f(1,0),f(1,0))$.
R_{Q_2} consists of the rules
$f(0,0) \to 0, \quad f(1,1) \to 0, \quad f(0,1) \to 1, \quad f(f(1,0),f(1,0)) \to 0$.
$NORM_2 = \{\, 0, 1, f(1,0), f(0,f(1,0)), f(1,f(1,0)), f(f(1,0),0), f(f(1,0),1)\,\}$.
Q_3 consists of the equations
$f(0,0) \approx 0, \quad f(1,1) \approx 0, \quad f(0,1) \approx 1, \quad 0 \approx f(f(1,0),f(1,0)),$
$0 \approx f(f(0,f(1,0)),f(0,f(1,0))), \quad 0 \approx f(f(1,f(1,0)),f(1,f(1,0)))$,
$0 \approx f(f(f(1,0),0),f(f(1,0),0)), \quad 0 \approx f(f(f(1,0),1),f(f(1,0),1))$.
R_{Q_3} consists of the rules
$f(0,0) \to 0, \quad f(1,1) \to 0, \quad f(0,1) \to 1, \quad f(f(1,0),f(1,0)) \to 0,$
$f(f(0,f(1,0)),f(0,f(1,0))) \to 0, \quad f(f(1,f(1,0)),f(1,f(1,0))) \to 0,$
$f(f(f(1,0),0),f(f(1,0),0)) \to 0, \quad f(f(f(1,0),1),f(f(1,0),1)) \to 0$.
Continuing in this manner we get that

$$R_{Q_i} \subset R_{Q_{i+1}} \text{ for } i \geq 1\,. \tag{13}$$

Let $R_{P_1 \cup Q_1} = R_{P_1}$, $R_{P_2 \cup Q_2} = R_{P_2}$, and $R_{P_3 \cup Q_3} = R_{P_3} \cup R_{Q_3}$.

Example 16. Let $\Sigma = \Sigma_0 \cup \Sigma_1$, $\Sigma_0 = \{ 0, 1 \}$, and $\Sigma_1 = \{ g, h \}$. Let TES E consist of the equations

$gx_1 \approx x_1, \quad hx_1 \approx hx_2$.

Let $p = 0$ and $q = 1$.

We now construct the GTESs P_1, P_2, and P_3. Then $NORM_0 = \{ 0, 1 \}$. P_1 consists of the equation $0 \approx g0$.
R_1 consists of the rule $g0 \to 0$.
$NORM_1 = \{ 0, 1, g1, h0, h1 \}$.
$P_2 = R_1$ and $R_2 = P_2$.
$NORM_2 = \{ 0, 1, g1, h0, h1, gg1, hg1, gh0, hh0, gh1, hh1 \}$.
$P_3 = R_2$ and $R_3 = P_3$.

We now construct the GTESs Q_1, Q_2, and Q_3. Then $NORM_0 = \{ 0, 1 \}$. Q_1 consists of the equation $1 \approx g1$.
R_{Q_1} consists of the rule $g1 \to 1$.
$NORM_1 = \{ 0, 1, g0, h0, h1 \}$.
$Q_2 = R_{Q_1}$ and $R_{Q_2} = Q_2$.
$NORM_2 = \{ 0, 1, g0, h0, h1, gg0, hg0, gh0, hh0, gh1, hh1 \}$.
$Q_3 = R_{Q_2}$ and $R_{Q_3} = Q_3$.
$R_{P_1} \cup R_{Q_1} = R_{P_1 \cup Q_1} = R_{P_2 \cup Q_2} = R_{P_3 \cup Q_3}$.

We get the following result by direct inspection of the definition of the GTES P_i and GTRS R_i, $i \geq 1$.

Statement 3. *For each* $i \geq 1$, $\leftrightarrow^*_{P_i} \subseteq \leftrightarrow^*_{P_{i+1}} \subseteq \leftrightarrow^*_E$.

We can show the following result similarly to Lemma 4.

Lemma 7. *For each* $i \geq 1$, *we can effectively construct the GTES* P_i.

Lemma 8. *For each* $i \geq 1$, $sub(p{\downarrow_{R_{P_i}}}) \cup (sub(R_{P_i}) - lhs(R_{P_i})) \cup \{ t{\downarrow_{R_{P_i}}} \mid height(t) \leq i \} \subseteq NORM_i$.

Proof. By induction on i.

□

We now present our semi-decision procedure.

Procedure *PRO4 Input:* A variable preserving TES E over the ranked alphabet Σ and ground terms $p, q \in T_\Sigma$.
Output: • 'yes' if $p \leftrightarrow^*_E q$,
• 'no' if $(p, q) \notin \leftrightarrow^*_E$ and the procedure halts,
• undefined if the procedure does not halt.
compute P_1, R_{P_1}, Q_1, R_{Q_1}, and $R_{P_1 \cup Q_1}$;
if $p{\downarrow_{R_{P_1 \cup Q_1}}} = q{\downarrow_{R_{P_1 \cup Q_1}}}$, then begin output 'yes'; halt end;
$i := 1$;
1: $i := i + 1$;
compute P_i, R_{P_i}, Q_i, R_{Q_i}, and $R_{P_i \cup Q_i}$;
if $p{\downarrow_{R_{P_i \cup Q_i}}} = q{\downarrow_{R_{P_i \cup Q_i}}}$, then begin output 'yes'; halt end;

if $i = 2$, then goto 1;
if $R_{P_{i-2}} = R_{P_{i-1}} = P_i$, or $R_{Q_{i-2}} = R_{Q_{i-1}} = Q_i$,
then begin output 'no'; halt end;
goto 1

Example 17. We continue Example 15. By Statement 3 and (11), $p{\downarrow_{R_{P_i \cup Q_i}}} \neq q{\downarrow_{R_{P_i \cup Q_i}}}$ for $i \geq 1$. Hence procedure *PRO4* does not output 'yes'. By (12) and (13), procedure *PRO4* does not output 'no'. Hence procedure *PRO4* does not output anything and does not halt at all.

Example 18. We continue Example 16. Observe that
$p{\downarrow_{R_{P_1 \cup Q_1}}} = 0 \neq 1 = q{\downarrow_{R_{P_1 \cup Q_1}}}$,
$p{\downarrow_{R_{P_2 \cup Q_2}}} = 0 \neq 1 = q{\downarrow_{R_{P_2 \cup Q_2}}}$,
$p{\downarrow_{R_{P_3 \cup Q_3}}} = 0 \neq 1 = q{\downarrow_{R_{P_3 \cup Q_3}}}$, and
$R_{P_1} = R_{P_2} = P_3$.
Hence procedure *PRO4* outputs 'no' and halts in the third step.

Example 19. Let $\Sigma = \Sigma_0 \cup \Sigma_1 \cup \Sigma_2$, $\Sigma_0 = \{\$, \#\}$, $\Sigma_1 = \{f, g\}$, $\Sigma_2 = \{h\}$. Consider the TES $E = \{ fffx_1 \approx gfx_1,\ h(x_1, x_1) \approx \$ \}$. As in Example 2, we can show that the basic Knuth-Bendix completion procedure runs forever on this example. Moreover, it is still open whether the goal-directed completion procedure halts on the TES E and any goal.

Let $n \geq 1$. Let $p = h(f^n\$, gf^{n-1}\$)$ and $q = \$$. We raise the problem whether $p \leftrightarrow_E^* q$. We now apply procedure *PRO4* to the TES E and the terms p, q.
GTRS R_{P_1} consists of the rules
$f^i\$ \to gf^{i-1}\$$ for $2 \leq i \leq n$,
$h(\$, \$) \to \$$,
$h(\#, \#) \to \$$.
GTRS R_{Q_1} consists of the rules
$h(\$, \$) \to \$$,
$h(\#, \#) \to \$$.
GTRS R_{P_2} consists of the rules
$f^2gf\$ \to gf\$$,
$h(\$, \$) \to \$$,
$h(\#, \#) \to \$$.
$h(f\$, f\$) \to \$$,
$h(f\#, f\#) \to \$$.
$h(g\$, g\$) \to \$$,
$h(g\#, g\#) \to \$$.
GTRS R_{Q_2} consists of the rules
$h(\$, \$) \to \$$,
$h(\#, \#) \to \$$,
$h(f\$, f\$) \to \$$,
$h(f\#, f\#) \to \$$.
$h(g\$, g\$) \to \$$,
$h(g\#, g\#) \to \$$.

Clearly,

$p{\downarrow_{R_{P_2\cup Q_2}}} = q{\downarrow_{R_{P_2\cup Q_2}}}$.

Hence procedure *PRO4* outputs 'yes' and halts in the second step.

We now show the correctness of Procedure *PRO4*.

Lemma 9. *Assume that* $R_{i-1} = R_i = P_{i+1}$ *and* $NORM_{i-1} \subset NORM_i$ *for some* $i \geq 2$. *Then for each equation* $l \approx r$ *of* $E \cup E^{-1}$ *with* $l \in T_\Sigma(X_{k+m})$, $r \in T_\Sigma(X_k \cup X_{[k+m+1,k+m+\ell]})$, $k, m \geq 0$, $\ell \geq 1$, *and for any* $u_1, \ldots, u_{k+m} \in sub(p{\downarrow_{R_i}}) \cup (sub(R_i) - lhs(R_i))$, R_i *does not reach* $p{\downarrow_{R_i}}$ *starting from* $l[u_1, \ldots, u_{k+m}]{\downarrow_{R_i}}$ *attached to some context.*

Proof. By contradiction. Assume that there is an equation $l \approx r$ of $E \cup E^{-1}$ with $l \in T_\Sigma(X_{k+m})$, $r \in T_\Sigma(X_k \cup X_{[k+m+1,k+m+\ell]})$, $k, m \geq 0$, $\ell \geq 1$, and there are $u_1, \ldots, u_{k+m} \in sub(p{\downarrow_{R_i}}) \cup (sub(R_i) - lhs(R_i))$ such that R_i reaches $p{\downarrow_{R_i}}$ starting from $l[u_1, \ldots, u_{k+m}]{\downarrow_{R_i}}$ attached to some context. By $R_i = P_{i+1}$, we do not put equations in P_{i+1} in item (b) of its definition. Consequently, for any v_{k+m+1}, …, $v_{k+m+\ell} \in NORM_i$,

$$l[u_1, \ldots, u_{k+m}]{\downarrow_{R_i}} = r[u_1, \ldots, u_m, v_{k+m+1}, \ldots, v_{k+m+\ell}]{\downarrow_{R_i}} \ .$$

Hence by our indirect assumption, R_i reaches $p{\downarrow_{R_i}}$ starting from $r[u_1, \ldots, u_m, v_{k+m+1}, \ldots, v_{k+m+\ell}]{\downarrow_{R_i}}$ attached to some context. Hence there is a $u \in C_\Sigma$ such that

$$u[r[u_1, \ldots, u_m, v_{k+m+1}, \ldots, v_{k+m+\ell}]{\downarrow_{R_i}}] \xrightarrow[R]{*} p{\downarrow_{R_i}} \ .$$

Then $u[r[u_1, \ldots, u_m, v_{k+m+1}{\downarrow_{R_i}}, v_{k+m+2}, \ldots, v_{k+m+\ell}]] \rightarrow^*_{R_i}$

$u[r[u_1, \ldots, u_m, v_{k+m+1}, \ldots, v_{k+m+\ell}]{\downarrow_{R_i}}] \rightarrow^*_{R_i} p{\downarrow_{R_i}}$. By Lemma 2, $v_{k+m+1}{\downarrow_{R_i}} \in sub(p{\downarrow_{R_i}}) \cup (sub(R_i) - lhs(R_i))$. Since $R_{i-1} = R_i$,

$v_{k+m+1}{\downarrow_{R_{i-1}}} \in sub(p{\downarrow_{R_{i-1}}}) \cup (sub(R_{i-1}) - lhs(R_{i-1})) \subseteq NORM_{i-1}$.

By definition, v_{k+m+1} is an arbitrary element of $NORM_i$. Consequently, we have $NORM_i \subseteq NORM_{i-1}$. This is a contradiction.

□

Lemma 10. *Let* $i \geq 2$. *If* $R_{i-1} = R_i = R_{i+1}$ *and* $NORM_{i-1} = NORM_i$, *then* $NORM_i = NORM_{i+1}$.

Proof. First we show that $NORM_i \subseteq NORM_{i+1}$. Let $s \in NORM_i$ be arbitrary. If $s \in sub(p{\downarrow_{R_i}}) \cup (sub(R_i) - lhs(R_i)) \cup \{\, t{\downarrow_{R_i}} \mid t \in NORM_{i-1} \,\}$, then $s \in sub(p{\downarrow_{R_{i+1}}}) \cup (sub(R_{i+1}) - lhs(R_{i+1})) \cup \{\, t{\downarrow_{R_{i+1}}} \mid t \in NORM_i \,\}$. Hence $t \in NORM_{i+1}$. If $s = f(t_1, \ldots, t_m){\downarrow_{R_i}}$ for some $f \in \Sigma_m$ and $t_1, \ldots, t_m \in NORM_{i-1}$, then $s = f(t_1, \ldots, t_m){\downarrow_{R_{i+1}}}$ with $f \in \Sigma_m$ and $t_1, \ldots, t_m \in NORM_i$. Hence $t \in NORM_{i+1}$.

We now show that $NORM_{i+1} \subseteq NORM_i$. Let $s \in NORM_{i+1}$ be arbitrary. If $s \in sub(p{\downarrow_{R_{i+1}}}) \cup (sub(R_{i+1}) - lhs(R_{i+1})) \cup \{\, t{\downarrow_{R_{i+1}}} \mid t \in NORM_i \,\}$, then $s \in sub(p{\downarrow_{R_i}}) \cup (sub(R_i) - lhs(R_i)) \cup \{\, t{\downarrow_{R_i}} \mid t \in NORM_{i-1} \,\}$. Hence $t \in NORM_i$.

If $s = f(t_1, \ldots, t_m)\downarrow_{R_{i+1}}$ for some $f \in \Sigma_m$ and $t_1, \ldots, t_m \in NORM_i$, then $s = f(t_1, \ldots, t_m)\downarrow_{R_i}$ for $f \in \Sigma_m$ and $t_1, \ldots, t_m \in NORM_{i-1}$. Hence $t \in NORM_i$.

□

Lemma 11. *For each $i \geq 2$, if $R_{i-1} = R_i = P_{i+1}$, then $R_i = R_{i+1} = P_{i+2}$.*

Proof. By the assumption $R_i = P_{i+1}$ and the definition of R_{i+1}, we have

$$R_i = R_{i+1}\,. \tag{14}$$

We now distinguish two cases.

Case 1: $NORM_{i-1} = NORM_i$. By Lemma 10,

$$NORM_i = NORM_{i+1}\,. \tag{15}$$

By (14) and (15), $P_{i+1} = P_{i+2}$. By the assumption $R_i = P_{i+1}$ and (14), we have $R_i = R_{i+1} = P_{i+2}$.

Case 2: $NORM_{i-1} \subset NORM_i$. Then by Lemma 9, for each equation $l \approx r$ of $E \cup E^{-1}$ with $l \in T_\Sigma(X_{k+m})$, $r \in T_\Sigma(X_k \cup X_{[k+m+1,k+m+\ell]})$, $k, m \geq 0$, $\ell \geq 1$, and for any $u_1, \ldots, u_{k+m} \in (sub(R_i) - lhs(R_i)) \cup sub(p\downarrow_{R_i})$, R_i does not reach $p\downarrow_{R_i}$ starting from $l[u_1, \ldots, u_{k+m}]\downarrow_{R_i}$ attached to some context. Then by (14), we do not put equations in P_{i+2} in item (b) in the definition of P_{i+2}. Hence $R_{i+1} = P_{i+2}$. By (14) the proof is complete.

□

Lemma 11 implies the following.

Lemma 12. *For each $i \geq 1$, if $R_{i-1} = R_i = P_{i+1}$, then for each $k \geq 1$, $R_i = R_{i+k} = P_{i+k+1}$.*

We now show the correctness of Procedure *PRO4*.

Lemma 13. *For any $n \geq 1$, $t_1, \ldots, t_n \in T_\Sigma$, if $p \leftrightarrow_E t_1 \leftrightarrow_E t_2 \leftrightarrow_E \cdots \leftrightarrow_E t_n$, then there is $i \geq 1$ such that $p \leftrightarrow^*_{P_i} t_1 \leftrightarrow^*_{P_i} t_2 \leftrightarrow^*_{P_i} \cdots \leftrightarrow^*_{P_i} t_n$.*

Proof. We proceed by induction on n.

Base Case: $n = 1$. Assume that $p \leftrightarrow_E t_1$. Then there is an equation $l \approx r$ of $E \cup E^{-1}$ with $l \in T_\Sigma(X_{k+m})$, $r \in T_\Sigma(X_{k+m+\ell})$, $k, m, \ell \geq 0$, and there is $u \in C_\Sigma$, $u_1, \ldots, u_{k+m}, v_{k+m+1}, \ldots, v_{k+m+\ell} \in T_\Sigma$ such that

$$p = u[l[u_1, \ldots, u_{k+m}]] \tag{16}$$

and $t_1 = u[r[u_1, \ldots, u_k, v_{k+m+1}, \ldots, v_{k+m+\ell}]]$.
Let $i = max\{\, height(v_{k+1}), \ldots, height(v_{k+m+\ell})\,\}$. By Lemma 8, $v_{k+m+1}\downarrow_{R_i}, \ldots, v_{k+m+\ell}\downarrow_{R_i}$ are in $NORM_i$. By (16), R_i reaches $p\downarrow_{R_i}$ from $l[u_1\downarrow_{R_i}, \ldots, u_{k+m}\downarrow_{R_i}]\downarrow_{R_i}$ attached to some context. By the definition of P_{i+1}, the equation

$$l[u_1\downarrow_{R_i}, \ldots, u_{k+m}\downarrow_{R_i}] \approx r[u_1\downarrow_{R_i}, \ldots, u_k\downarrow_{R_i}, v_{k+m+1}\downarrow_{R_i}, \ldots, v_{k+m+\ell}\downarrow_{R_i}]$$

is in $\leftrightarrow^*_{P_i}$ or P_{i+1}. Hence, by the definition of R_i and Statement 3,

$p = u[l[u_1, \ldots, u_{k+m}]] \leftrightarrow^*_{P_{i+1}} u[l[u_1{\downarrow_{R_i}}, \ldots, u_{k+m}{\downarrow_{R_i}}]] \leftrightarrow^*_{P_{i+1}}$
$u[r[u_1{\downarrow_{R_i}}, \ldots, u_k{\downarrow_{R_i}}, v_{k+m+1}{\downarrow_{R_i}}, \ldots, v_{k+m+\ell}{\downarrow_{R_i}}]{\downarrow_{R_i}}] \leftrightarrow^*_{P_i}$
$u[r[u_1, \ldots, u_k, v_{k+m+1}, \ldots, v_{k+m+\ell}]] = t_1$.
Then we have $p \leftrightarrow^*_{P_{i+1}} t_1$.

Induction Step: Let $n \geq 1$, and assume that the satement holds for $1, 2, \ldots, n$. We now show that the satement holds for $n+1$. To this end, assume that

$$p \underset{E}{\leftrightarrow} t_1 \underset{E}{\leftrightarrow} t_2 \underset{E}{\leftrightarrow} \cdots \underset{E}{\leftrightarrow} t_{n+1} \,. \tag{17}$$

By the induction hypothesis, there is $j \geq 1$ such that

$$p \overset{*}{\underset{P_j}{\leftrightarrow}} t_1 \overset{*}{\underset{P_j}{\leftrightarrow}} t_2 \overset{*}{\underset{P_j}{\leftrightarrow}} \cdots \overset{*}{\underset{P_j}{\leftrightarrow}} t_n \,. \tag{18}$$

Hence

$$t_n \overset{*}{\underset{R_i}{\rightarrow}} p{\downarrow_{R_i}} \ . \tag{19}$$

By (17), there is an equation $l \approx r$ in $E \cup E^{-1}$ with $l \in T_\Sigma(X_{k+m})$, $r \in T_\Sigma(X_k \cup X_{[k+m+1,k+m+\ell]})$ for some $k, m, \ell \geq 0$, and there are $u \in C_\Sigma$,
$u_1, \ldots, u_{k+m}, v_{k+m+1}, \ldots, v_{k+m+\ell} \in T_\Sigma$ such that

$$t_n = u[l[u_1, \ldots, u_{k+m}]] \text{ and } t_{n+1} = u[r[u_1, \ldots, u_k, v_{k+m+1}, \ldots, v_{k+m+\ell}]] \,. \tag{20}$$

Let $i = max\{\, j, height(v_{k+m+1}), \ldots, height(v_{k+m+\ell}) \,\}$. By Lemma 8, $v_{k+m+1}{\downarrow_{R_i}}$, $\ldots$, $v_{k+m+\ell}{\downarrow_{R_i}}$ are in $NORM_i$. Clearly, $l[u_1{\downarrow_{R_i}}, \ldots, u_{k+m}{\downarrow_{R_i}}] \rightarrow^*_{R_i} l[u_1{\downarrow_{R_i}}, \ldots, u_{k+m}{\downarrow_{R_i}}]{\downarrow_{R_i}}$. Then by (19) and (20), R_i reaches $p{\downarrow_{R_i}}$ starting from $l[u_1{\downarrow_{R_i}}, \ldots, u_{k+m}{\downarrow_{R_i}}]{\downarrow_{R_i}}$ attached to some context. By the definition of P_{i+1}, the equation
$l[u_1{\downarrow_{R_i}}, \ldots, u_{k+m}{\downarrow_{R_i}}] \approx r[u_1{\downarrow_{R_i}}, \ldots, u_k{\downarrow_{R_i}}, v_{k+m+1}{\downarrow_{R_i}}, \ldots, v_{k+m+\ell}{\downarrow_{R_i}}]$
is in $\leftrightarrow^*_{P_i}$ or P_{i+1}. Hence, by the definition of R_i and Statement 3,
$t_n = u[l[u_1, \ldots, u_{k+m}]] \leftrightarrow^*_{P_{i+1}} u[l[u_1{\downarrow_{R_i}}, \ldots, u_{k+m}{\downarrow_{R_i}}]] \leftrightarrow_{P_{i+1}}$
$u[r[u_1{\downarrow_{R_i}}, \ldots, u_k{\downarrow_{R_i}}, v_{k+m+1}{\downarrow_{R_i}}, \ldots, v_{k+m+\ell}{\downarrow_{R_i}}]] \leftrightarrow^*_{P_{i+1}}$
$u[r[u_1, \ldots, u_k, v_{k+m+1}, \ldots, v_{k+m+\ell}]] = t_{n+1}$.
By (18), $p \leftrightarrow^*_{P_{i+1}} t_1 \leftrightarrow^*_{P_{i+1}} t_2 \leftrightarrow^*_{P_{i+1}} \cdots \leftrightarrow^*_{P_{i+1}} t_n \leftrightarrow^*_{P_{i+1}} t_{n+1}$.

□

By Statement 3, Lemma 12, and Lemma 13 we have the following result.

Lemma 14. *For each $i \geq 2$, if $R_{i-1} = R_i = P_{i+1}$, then for each $q' \in T_\Sigma$, $p \leftrightarrow^*_{P_i} q'$ if and only if $p \leftrightarrow^*_E q'$.*

We can show the following in the same way as Theorem 1.

Theorem 5. *If $p \leftrightarrow^*_E q$, then procedure PRO4 outputs 'yes' and halts.*

We can show the following in the same way as Theorem 2.

Theorem 6. *If procedure PRO4 outputs 'yes' and halts, then $p \leftrightarrow^*_E q$. If procedure PRO4 outputs 'no' and halts, then $(p, q) \notin \leftrightarrow^*_E$.*

Theorems 5 and 6 imply the following.

Theorem 7. *If $p \leftrightarrow^*_E q$, then procedure PRO4 outputs 'yes' and halts. Otherwise, either PRO4 outputs 'no' and halts, or PRO4 does not halt at all.*

7 Comparison with the Knuth-Bendix completion procedure

We now compare procedures *PRO3* and *PRO4* with the basic Knuth-Bendix completion procedure (see Section 7.1 in [1]), the improved version of the Knuth-Bendix completion procedure described by a set of inference rules (see Section 7.2 in [1]), the goal-directed completion procedure based on SOUR graphs [13, 14], and the unfailing Knuth-Bendix completion procedure [2]. In contrast to all versions of the Knuth-Bendix procedure, Procedures *PRO3* and *PRO4* do not compute any critical pairs and do not use a reduction order. They do not attempt to construct a convergent TRS equivalent to E. When *PRO3* and *PRO4* run a congruence closure algorithm for the TES E over the subterm graph of E [4, 15], they compute and then process only finitely many ground instances $(\overline{s}, \overline{t})$ of finitely many elements (s, t) of the relation $\leftrightarrow_E^*$, where s, t may contain variables. Here (s, t) need not be a critical pair computed by the basic Knuth-Bendix completion procedure. In fact, the ground instances $(\overline{s}, \overline{t})$ are elements of the equivalence relation $\leftrightarrow_E^* \cap (sub(E) \times sub(E))$. Procedures *PRO3* and *PRO4* compute a representative r of $\overline{s}$ and $\overline{t}$ for the equivalence relation $\leftrightarrow_E^* \cap (sub(E) \times sub(E))$. The representative r becomes the normal form of $\overline{s}$ and $\overline{t}$ for the rewrite relation induced by the constructed reduced GTRS. Hence, *PRO3* and *PRO4* do not compare the normal forms of s and t via any reduction order. In contrast, the basic Knuth-Bendix completion procedure reduces the terms in each critical pair to their normal forms. Then tries to orient the normal forms into a rewrite rule. In this way the procedure orients all instances of these terms as well. The improved version of the Knuth-Bendix completion procedure described by a set of inference rules (see Section 7.2 in [1]) also processes each critical pair and also orients the obtained pair, and hence all of its instances. The unfailing Knuth-Bendix completion procedure [2] applies orientable instances of equations in E with respect to a reduction order $>$.

To illustrate the efficiency of the goal-directed completion procedure, Lynch [13] presented the following example. Let the ranked alphabet Σ consist of the unary symbols f, g and the nullary symbols \$, #. Consider the variable preserving TES $E = \{ fffx \approx gfx \}$. We raise the problem whether $\$ \leftrightarrow_E^* \#$. On the one hand, the basic Knuth-Bendix completion procedure runs forever on this example [13]. On the other hand, the goal-directed completion procedure does not generate any rule applicable to \$ or #. Therefore, the goal-directed completion procedure outputs 'no' and halts [13]. Lynch and Strogova [14] said that "the goal-directed completion procedure compiles the TES E and the goal (p, q). After the compilation is finished, we cannot apply a schematization of an equation in the completed system. Therefore, the goal-directed completion procedure outputs 'no' and halts. This is an example where the goal-directed completion procedure is superior to the basic Knuth-Bendix algorithm." It is still open whether the goal-directed completion procedure halts on the TES E and any goal [13]. As for the above example, *PRO3* gives the correct answer and then halts on the TES E and any terms $p, q \in T_\Sigma$.

We conjecture that there are variable preserving TES E and ground terms p, q

such that Conditions (a)-(c) hold.

(a) The basic Knuth-Bendix completion procedure runs forever on E.

(b) There is a goal (p, q) such that the goal-directed completion procedure does not stop on E and (p, q).

(c) Procedure *PRO3* gives the correct answer and then halts on the TES E and any terms $p, q \in T_\Sigma$.

Let TES E be as in Example 11. We conjecture that there is $q \in T_\Sigma$ such that the symbols a, c do not appear in q and that the goal-directed completion procedure does not halt on the TES E and the goal $(aaa\$, q)$. On the other hand, let $q \in T_\Sigma$ be arbitrary such that the symbols a, c do not appear in q. On the input E, $aaa\$$, q, Procedure *PRO3* outputs 'no', the correct answer, and then halts, see Example 11.

Procedures *PRO3* and *PRO4* attempt to construct the reduced GTRSs R_P and R_Q, rather than a convergent term rewrite system equivalent to E, such that

- $R_P \cup R_Q \subseteq \leftrightarrow_E^*$,
- $p \leftrightarrow_{R_P}^* q$ or $\leftrightarrow_{R_P}^* \cap(\{\, p \,\} \times T_\Sigma) = \leftrightarrow_E^* \cap(\{\, p \,\} \times T_\Sigma)$, and
- $p \leftrightarrow_{R_Q}^* q$ or $\leftrightarrow_{R_Q}^* \cap(\{\, q \,\} \times T_\Sigma) = \leftrightarrow_E^* \cap(\{\, q \,\} \times T_\Sigma)$.

Thus R_P and R_Q need not be equivalent to E. By contrast, all versions of the Knuth-Bendix completion procedure attempt to transform a given TES E into an equivalent convergent term rewrite sytem. Since Snyder's ground completion algorithm does not apply orderings, procedures *PRO3* and *PRO4* do not apply any orderings as well.

We now present three examples where procedures *PRO3* and *PRO4* compute efficiently, probably more efficiently than all versions of the Knuth-Bendix completion procedure.

Example 20. [8, 16] Gallier et al [8] and Plaisted and Sattler-Klein [16] presented the following problem to illustrate that reducing a ground term to its normal form can take exponential time if a proper strategy is not used. Let $\Sigma = \Sigma_0 \cup \Sigma_1$, $\Sigma_0 = \{\, \$ \,\}$, and $\Sigma_1 = \{\, f, g \,\}$. Let $n \geq 2$. Let the GTRS R consist of the following rules:

$f\$ \to g\$$,
$fg\$ \to gf\$$,
$fg^2\$ \to gf^2\$$,
$\ldots$
$fg^n\$ \to gf^n\$$.

Plaisted and Sattler-Klein observed the following on page 156 in [16]. Although GTRS R is convergent, the right-hand sides can be further rewritten. An unskilful choice of rewrites can lead to an exponential time of process. The straightforward reduction of the term $gf^n\$$ can take a number of rewrite steps exponential in n. However, if we apply the rules in order of size, smallest first, to all other rules, the whole TRS can be rewritten to a reduced GTRS in a polynomial number of steps.

We form the TES E by adding the equation

$$fg^{n+1}x \approx gf^{n+1}x$$

to the set R. We now run procedure *PRO3* on the variable preserving TES E and the ground terms $p = f^{n+2}\$$ and $q = g^{n+2}\$$. Then

$\{ f\$ \approx g\$ \} = P_1 = R_{P_1} = Q_1 = R_{Q_1}$, $p{\downarrow_{R_{P_1 \cup Q_1}}} \neq q{\downarrow_{R_{P_1 \cup Q_1}}} = q$.

$R_1 \cup \{ fg\$ \approx g^2\$ \} = P_2 = R_{P_2} = Q_2 = R_{Q_2}$, $p{\downarrow_{R_{P_2 \cup Q_2}}} \neq q{\downarrow_{R_{P_2 \cup Q_2}}} = q$.

$R_2 \cup \{ fg^2\$ \approx g^3\$ \} = P_3 = R_{P_3} = Q_3 = R_{Q_3}$, $p{\downarrow_{R_{P_3 \cup Q_3}}} \neq q{\downarrow_{R_{P_3 \cup Q_3}}} = q$.

...

$R_{P_n} \cup \{ fg^n\$ \approx g^n\$ \} = P_n = R_{P_n} = Q_n = R_{Q_n}$, $p{\downarrow_{R_{P_{n+1} \cup Q_{n+1}}}} \neq q{\downarrow_{R_{P_{n+1} \cup Q_{n+1}}}}$.

$R_{P_{n+1}} \cup \{ fg^{n+1}\$ \approx g^{n+1}\$ \} = P_{n+2} = R_{P_{n+2}} = Q_{n+2} = R_{Q_{n+2}}$.

$P_{n+2} = R_{P_{n+3}} = Q_{n+2} = R_{Q_{n+3}}$.

Observe that $p{\downarrow_{R_{P_{n+2} \cup Q_{n+2}}}} = q{\downarrow_{R_{P_{n+2} \cup Q_{n+2}}}} = q$. Hence procedure *PRO3* outputs 'yes' and halts in the $(n+2)$nd step. The number of computation steps is polynomial. It should be clear that for all ground terms p and q, *PRO3* halts. It outputs 'yes' if $p \leftrightarrow_E^* q$. Otherwise it outputs 'no'.

Consider the lexicographic path order $>_{lpo}$ induced by the order $f > g > \$$ [1]. We now run the basic Knuth-Bendix completion procedure on the TES E and the reduction order $>_{lpo}$. In the initialization phase, the basic Knuth-Bendix completion procedure orients the equations of E. We obtain the TRS S consisting of the following rules:

$f\$ \to g\$$,
$fg\$ \to gf\$$,
$fg^2\$ \to gf^2\$$,
...
$fg^n\$ \to gf^n\$$,
$fg^{n+1}x \to gf^{n+1}x$.

Similarly to the first part of the example we have the following. The TRS S has no critical pairs. Hence the basic Knuth-Bendix procedure outputs S. The straightforward reduction of the term $f^{n+2}\$$ to $g^{n+2}\$$ by S takes a number of rewrite steps exponential in n. The improved Knuth-Bendix completion procedure reduces the right-hand sides of the first n rules as in the first part of the example. We obtain the TRS S' consisting of the following rules:

$f\$ \to g\$$,
$fg\$ \to gf\$$, $fg\$ \to gg\$$,
$fg^2\$ \to gf^2\$$, $fg^2\$ \to gfg\$$, $fg^2\$ \to g^3\$$,
...
$fg^n\$ \to gf^n\$$, $fg^n\$ \to gf^{n-1}g\$$, ..., $fg^n\$ \to g^{n+1}\$$,
$fg^{n+1}x \to gf^{n+1}x$.

In the best case, the reduction of the term $f^{n+2}\$$ to $g^{n+2}\$$ applies the rules

$f\$ \to g\$$,
$fg\$ \to gg\$$,
$fg^2\$ \to g^3\$$,
...
$fg^n\$ \to g^{n+1}\$$,
$fg^{n+1}x \to gf^{n+1}x$.

In the worst case, S' applies only the rules of S in the reduction of the term $f^{n+2}\$$ to $g^{n+2}\$$. Hence it takes a number of rewrite steps exponential in n as in the first part of the example. The goal-directed completion procedure computes fast on E and the goal (p, q). For experimental results, see the line of the problem Counter5 in Table 1 in Section 7 in [14].

Example 21. We now modify an example of Plaisted and Sattler-Klein [16] and Lynch and Strogova [14].

Let $n \geq 2$, $\Sigma = \Sigma_0 \cup \Sigma_2$, $\Sigma_0 = \{ \$_1, \$_2, \ldots, \$_n, \#_1, \#_2, \ldots, \#_n \}$, and $\Sigma_2 = \{ f, g \}$. Let the TES E consist of the following equations:

$f(\flat, \flat) \approx f(\#_0, \$_0)$,
$\$_0 \approx f(\$_1, \#_1)$,
$\#_0 \approx g(\#_1, \$_1)$,
$\$_1 \approx f(\$_2, \#_2)$,
$\#_1 \approx g(\#_2, \$_2)$,
$\ldots$
$\$_{n-1} \approx f(\$_n, \#_n)$,
$\#_{n-1} \approx g(\#_n, \$_n)$,
$\$_n \approx \#_n$,
$f(x_1, x_1) \approx g(x_1, x_1)$.

We now run procedure *PRO3* on the variable preserving TES E and the ground terms $p = f(\$_0, \#_0)$ and $q = g(\#_0, \#_0)$. Then

$\{ f(\$_1, \#_1) \approx \$_0,\ g(\#_1, \$_1) \approx \#_0 \} = P_1 = R_{P_1}$,
$\{ g(\#_1, \$_1) \approx \#_0,\ f(\#_0, \#_0) \approx g(\#_0, \#_0) \} = Q_1 = R_{Q_1}$,
$R_{P_1} \cup \{ f(\$_2, \#_2) \approx \$_1,\ g(\#_2, \$_2) \approx \$_1 \} = P_2 = R_{P_2}$,
$R_{Q_1} \cup \{ f(\$_1, \#_1) \approx \$_0,\ f(\$_2, \#_2) \approx \$_1,\ g(\#_2, \$_2) \approx \#_1 \} = Q_2 = R_{Q_2}$,
$\ldots$
$R_{P_{n-1}} \cup \{ f(\$_n, \#_n) \approx \$_{n-1},\ g(\#_n, \$_n) \approx \#_{n-1} \} = P_n = R_{P_n}$,
$R_{Q_{n-1}} \cup \{ f(\$_{n-1}, \#_{n-1}) \approx \$_{n-2},\ g(\#_n, \$_n) \approx \#_{n-1} \} = Q_n = R_{Q_n}$.

R_{P_n} consists of the following rules:

$f(\$_1, \#_1) \to \$_0$,
$g(\#_1, \$_1) \to \#_0$,
$f(\$_2, \#_2) \to \$_1$,
$g(\#_2, \$_2) \to \#_1$,
$\ldots$
$f(\$_n, \#_n) \to \$_{n-1}$,
$g(\#_n, \$_n) \to \#_{n-1}$,
$R_{P_n} \cup \{ \$_n \approx \#_n \} = P_{n+1}$.

$R_{P_{n+1}}$ consists of the following rules:

$f(\$_0, \#_0) \to \$_0$,
$f(\$_1, \$_1) \to \$_0$,
$f(\$_2, \$_2) \to \$_1$,
$\ldots$
$g(\$_n, \$_n) \to \$_{n-1}$,
$\#_0 \to \$_0$,

$\#_1 \to \$_1$,
$\#_2 \to \$_2$,
$\ldots$
$\#_n \to \$_n$.
$R_{Q_n} \cup \{\, f(\$_n, \#_n) \approx \$_{n-1},\ \$ \approx \# \,\} = Q_{n+1}$,
$R_{Q_{n+1}} = R_{P_{n+1}} \cup \{\, f(\#_0, \#_0) \to g(\#_0, \#_0) \,\}$.
$P_{n+2} = R_{P_{n+3}} = Q_{n+2} = R_{Q_{n+3}}$.

Clearly, $p{\downarrow_{R_{P_{n+1}}}} = q{\downarrow_{R_{P_{n+1}}}}$. Consequently, procedure *PRO3* outputs 'yes' and halts in the $(n+1)$st step. The number of computation steps is polynomial.

Consider the lexicographic path order $>_{lpo}$ induced by the order

$$\flat > \$_0 > \$_1 > \cdots > \$_n > \#_0 > \#_1 > \cdots > \#_n > f > g.$$

We now run the basic Knuth-Bendix completion procedure on the TES E and the reduction order $>_{lpo}$. In the initialization phase, the basic Knuth-Bendix completion procedure orients the equations of E. We obtain the TRS S consisting of the following rules:

$\$_0 \to f(\$_1, \#_1)$,
$\#_0 \to g(\#_1, \$_1)$,
$\$_1 \to f(\$_2, \#_2)$,
$\#_1 \to g(\#_2, \$_2)$,
$\ldots$
$\$_{n-1} \to f(\$_n, \#_n)$,
$\#_{n-1} \to g(\#_n, \$_n)$,
$\$_n \to \#_n$,
$f(\flat, \flat) \to f(\#_0, \$_0)$,
$f(x_1, x_1) \to g(x_1, x_1)$.

The last two rules yield the critical pair $\langle f(\#_0, \$_0), g(\flat, \flat)\rangle$. Observe that $f(\#_0, \$_0)$ has a unique $\to_S$ normal form, and that $size(f(\#_0, \$_0){\downarrow_S}) = 2^{n+1}$. Thus the completed system contains a rule with a left-hand side of size 2^{n+1}. The improved Knuth-Bendix completion procedure also yields the TRS S and the above critical pair. Again, the completed system contains a rule with a left-hand side of size 2^{n+1}. The goal-directed completion procedure based on SOUR graphs [13, 14] stores the term $f(\#_0, \$_0){\downarrow_S}$ in linear space in n.

Example 22. Let $\Sigma = \Sigma_0 \cup \Sigma_1$, $\Sigma_0 = \{\, \$ \,\}$, and $\Sigma_1 = \{\, a, b \,\}$. Let the GTES F consist of the equation $abbax_1 \approx x_1$. Furthermore, let the GTES E consist of the equations

$abbax_1 \approx x_1$, $a\$ \approx \$$, $b\$ \approx \$$.

It is well-known that there is no convergent TRS R equivalent to F, see Theorem 4.2.18 in [10]. Hence there is no convergent TRS R equivalent to E either. Consequently, the basic Knuth-Bendix completion procedure (see Section 7.1 in [1]), the improved version of the Knuth-Bendix completion procedure described by a set of inference rules (see Section 7.2 in [1]) cannot produce a convergent TRS R equivalent to E.

Let $p, q \in T_\Sigma$ be arbitrary. First, we run the procedure *PRO3* on the input E, p, q. Procedure *PRO3* outputs 'yes' and halts in the first or second step. The resulting reduced GTRS is a subset of

$\{ a\$ \to \$, b\$ \to \$ \}$.

Second, we run the goal-directed completion procedure on the input E, (p, q). It computes all critical pairs and then processes them. Then it applies the resulting rules. The goal-directed completion procedure takes more time on E and the goal (p, q) than procedure *PRO3* on the input E, p, q.

8 Conclusion

We recalled the well known trivial semi-decision procedure *PRO1* for the ground word problem of variable preserving TESs and its straightforward generalization, the trivial semi-decision procedure *PRO2* for the ground word problem of TESs. On the basis of *PRO1*, we gave the semi-decision procedure *PRO3* for the ground word problem of variable preserving TESs. We gave examples when procedure *PRO3* was more efficient than procedure *PRO1*. Then we presented the semi-decision procedure *PRO4* for the ground word problem of term equation systems. We obtained it generalizing *PRO3* taking into account *PRO2*. We showed the correctness of *PRO3* and *PRO4*. We compared the procedures *PRO3* and *PRO4* with the basic Knuth-Bendix completion procedure and the goal-directed completion procedure based on SOUR graphs [13, 14].

Procedures *PRO3* or *PRO4* compute in a different way than all versions of the Knuth-Bendix completion procedure. To some instances of the ground word problem of a TES E, they give an answer sooner than all versions of the Knuth-Bendix completion procedure or it is open whether some version of the Knuth-Bendix completion procedure gives an answer at all. Assume that, given a TES E and ground terms p, q, we want to decide whether $p \leftrightarrow_E^* q$. The ground word problem is undecidable even for variable-preserving TESs. Consequently, we have no upper bound on the running time of any type of the Knuth-Bendix completion procedure on the input TES E any reduction order $>$ and the ground terms p, q. However, we assume beforehand that the basic Knuth-Bendix completion procedure or the goal-directed completion procedure or the nonfailing Knuth-Bendix completion procedure will stop on E, $>$, and p, q, and estimate its running time. We base our time estimate on the size of the input and the experimental results by the various implementations [7, 9, 12, 20] of all versions of the Knuth-Bendix completion procedure on inputs of similar size. Then we carry out the following steps. Simultaneously, we start all implementations of all versions of the Knuth-Bendix completion procedure on E and p, q. We wait for the estimated running time. If none of the procedures stop within this time, then they do not stop at all, or we underestimated the running time. Then we start the procedure *PRO3* or *PRO4* depending on whether TES E is variable preserving. In some cases *PRO3* or *PRO4* might give an answer sooner than all implementations of all versions of the Knuth-Bendix completion procedure.

We presented ad hoc examples when procedure *PRO3* was probably more ef-

ficient than the goal-directed completion procedure [13, 14]. However, to justify the introduction of procedures *PRO3* and *PRO4*, we need further evidence for the efficiency of the procedures *PRO3* and *PRO4*. We should present implementation results and theoretical arguments. We now raise questions on the efficiency of *PRO3* and *PRO4* compared to the various versions of the Knuth-Bendix completion procedure.

Question 1. *Is it true that for most instances of the ground word problem of a TES E, a correctly chosen version of the Knuth-Bendix completion procedure is more efficient than PRO3 or PRO4?*

Question 2. *For which instances of the ground word problem of a TES E, is a correctly chosen version of the Knuth-Bendix completion procedure more efficient than PRO3 or PRO4?*

Question 3. *Is it decidable for an instance of the ground word problem of a TES E, whether a correctly chosen version of the Knuth-Bendix completion procedure is more efficient than PRO3 or PRO4?*

Question 4. *Is there an instance of the ground word problem of a TES E, such that no version of the Knuth-Bendix completion procedure halts, and PRO3 or PRO4 halts?*

We can reduce an instance of the word problem for a TES E to an instance of the ground word problem for E over a larger alphabet Δ. Let E be a TES and p, q arbitrary terms over a ranked alphabet Σ. Assume that exactly the variables $x_1, \ldots, x_m$ appear in p or q. We now define the ranked alphabet Δ. It contains each element of Σ. Furthermore, for each $i = 1, \ldots, m$, we add a new constant $\#_i$ to Δ. We define p' from p and q' from q by replacing each occurrence of x_i with $\#_i$ for $i = 1, \ldots, m$. Then $p \leftrightarrow_E^* q$ over Σ if and only if $p' \leftrightarrow_E^* q'$ over Δ. Thus if we can decide whether $p' \leftrightarrow_E^* q'$ over Δ, then we can also decide whether $p \leftrightarrow_E^* q$ over Σ.

References

[1] F. Baader and T. Nipkow Term Rewriting and All That, Cambridge University Press, Cambridge, United Kingdom, 1998.

[2] L. Bachmair, N. Dershowitz, D. A. Plaisted, Completion without failure. In Resolution of equations in algebraic structures, Vol. 2, Rewriting techniques. Edited by H. Ait-Kaci and M. Nivat. pp. 130, Academic Press, Boston, MA, 1989,

[3] G. C. Ceitin: Associative calculus with an unsolvable equivalence problem. Tr. Mat. Inst. Akad. Nauk 52, 172-189 (1958) (in Russian).

[4] P. J. Downey, R. Sethi, and R. E. Tarjan: Variations on the Common Subexpression Problem. Journal of the ACM, 27 (1980) 758-771.

[5] Z. Fülöp and S. Vágvölgyi, Ground term rewriting rules for the word problem of ground term equations, Bulletin of the EATCS, **45** (1991) 186-201.

[6] Z. Fülöp and S. Vágvölgyi, Minimal Equational Representations of Recognizable Tree Languages, *Acta Informatica*, **34** (1997) 59-84.

[7] J.-M. Gaillourdet, T. Hillenbrand, B. Löchner, H. Spies: The New WALDMEISTER Loop at Work, in Franz Baader (Ed.): Automated Deduction - CADE-19, 19th International Conference on Automated Deduction, Proceedings. Lecture Notes in Computer Science 2741 Springer 2003, I 317-321.

[8] J. Gallier, P. Narendran, D. Plaisted, S. Raatz, and W. Snyder, An Algorithm for Finding Canonical Sets of Ground Rewrite Rules in Polynomial Time, *Journal of the Association for Computing Machinery*, **40** (1993) 1-16.

[9] T. Hillenbrand, Citius altius fortius: Lessons learned from the Theorem Prover WALDMEISTER, Electronic Notes in Theoretical Computer Science 86(1) (2003).

[10] M. Jantzen, Confluent string rewriting, Springer Verlag, Berlin 1988.

[11] Y. Matiyasevich, G. Sénizergues, Decision Problems for Semi-Thue Systems with a Few Rules, Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, 27-30 July 1996. IEEE Computer Society Press, 523-531.

[12] Serge Mechveliani, From a Computer Algebra Library to a System with an Equational Prover, in Bruno Buchberger, John A. Campbell (Eds.): Artificial Intelligence and Symbolic Computation, 7th International Conference, AISC 2004, Linz, Austria, September 22-24, 2004, Proceedings. Lecture Notes in Computer Science 3249 Springer 2004, 281-284.

[13] C. Lynch, Goal-Directed Completion Using SOUR Graphs, in Hubert Comon (Ed.): Rewriting Techniques and Applications, 8th International Conference, RTA-97, Proceedings. Lecture Notes in Computer Science 1232 Springer 1997, 8-22.

[14] C. Lynch, P. Strogova, SOUR graphs for efficient completion, Discrete Mathematics & Theoretical Computer Science **2** (1998) 1-25.

[15]] G. Nelson, D. C. Oppen: Fast Decision Procedures Based on Congruence Closure. J. ACM **27** (1980) 356-364

[16] D. Plaisted and A. Sattler-Klein, Proof lengths for equational completion, Information and Computation, **125** (1996) 154-170.

[17] S. Vágvölgyi, A fast algorithm for constructing a tree automaton recognizing a congruential tree language, Theoret. Comput. Sci, **115** (1993) 391-399.

[18] S. Vágvölgyi, Ground term rewriting. Bull. Eur. Assoc. Theor. Comput. Sci. EATCS No. 102 (2010), 153190.

[19] W. Snyder, A Fast Algorithm for Generating Reduced Ground Rewriting Systems from a set of Ground Equations, Journal of Symbolic Computation, 15 (1993) 415-450.

[20] I. Wehrman, A. Stump, E. M. Westbrook: Slothrop: Knuth-Bendix Completion with a Modern Termination Checker, Frank Pfenning (Ed.): Term Rewriting and Applications, 17th International Conference, RTA 2006, Proceedings. Lecture Notes in Computer Science 4098 Springer 2006, 287-296.

Received 5th June 2015

CONTENTS

ISSN 0324—721 X

Felelős szerkesztő és kiadó: Csirik János